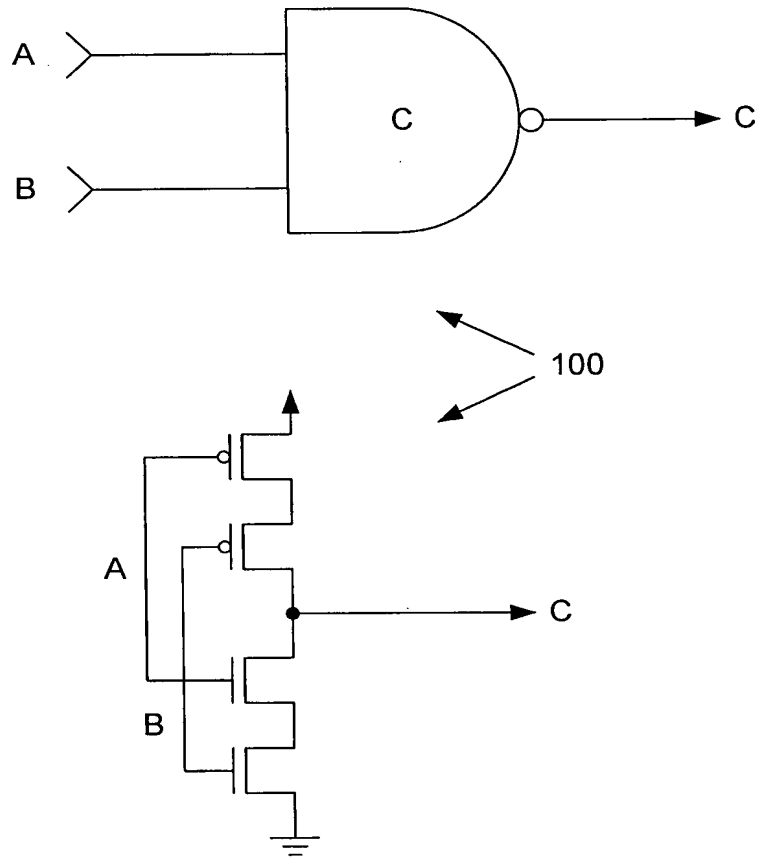


Fig. 1

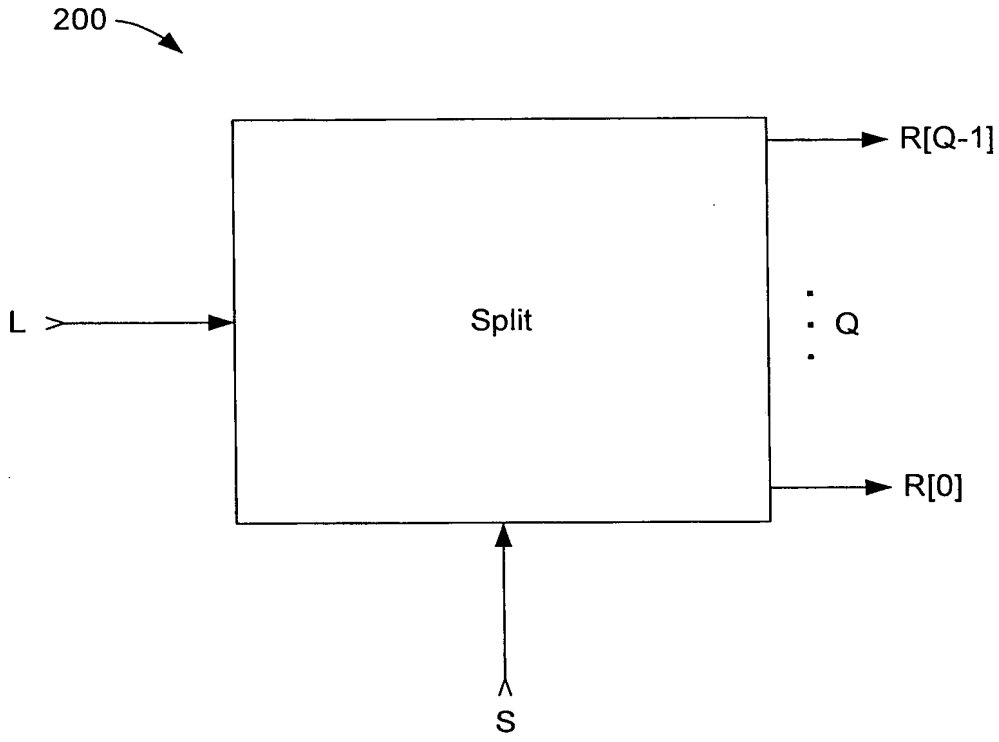


Fig. 2

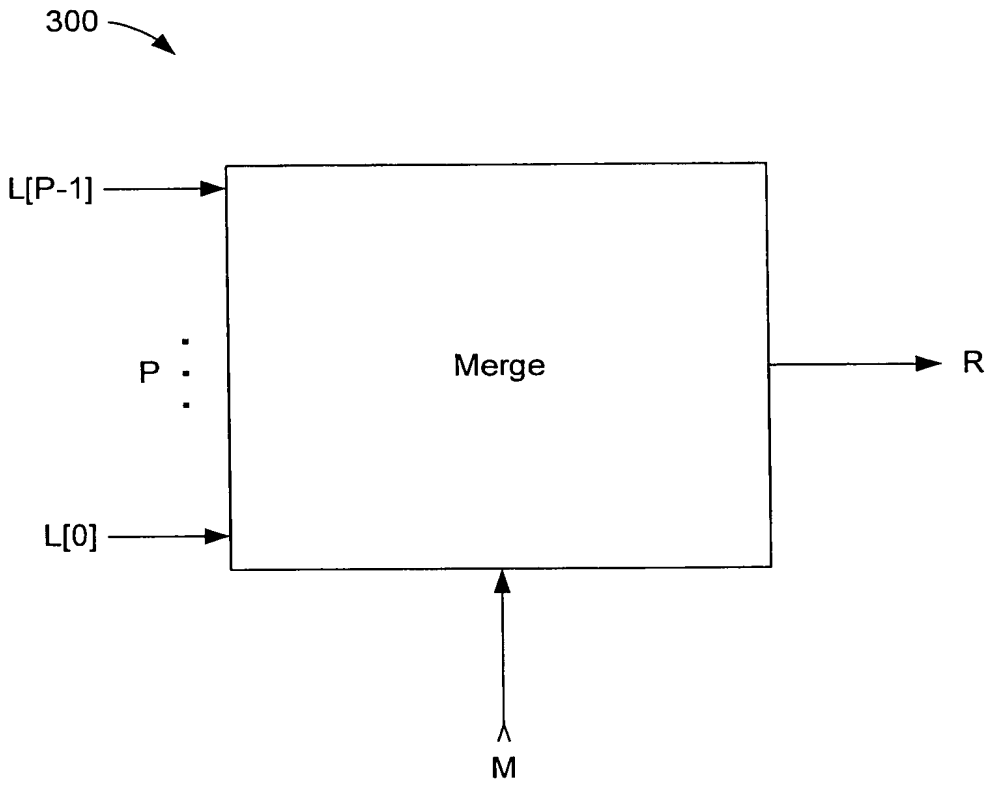


Fig. 3

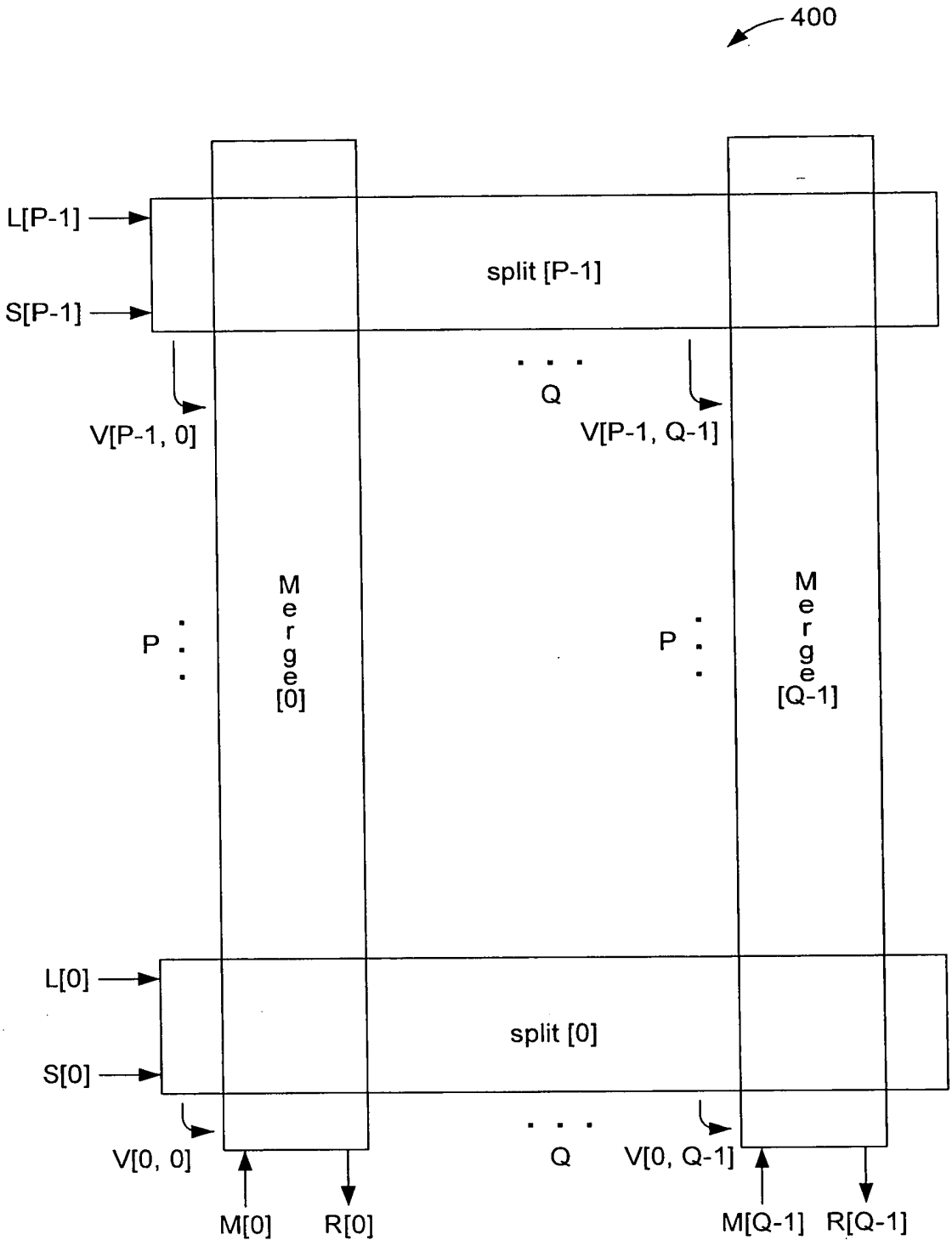

400
L[P-1]
S[P-1]
split [P-1]
Q
V[P-1, 0]
V[P-1, Q-1]
P
Merge [0]
P
Merge [Q-1]
L[0]
S[0]
split [0]
Q
V[0, 0]
V[0, Q-1]
M[0]
R[0]
M[Q-1]
R[Q-1]

Fig. 4

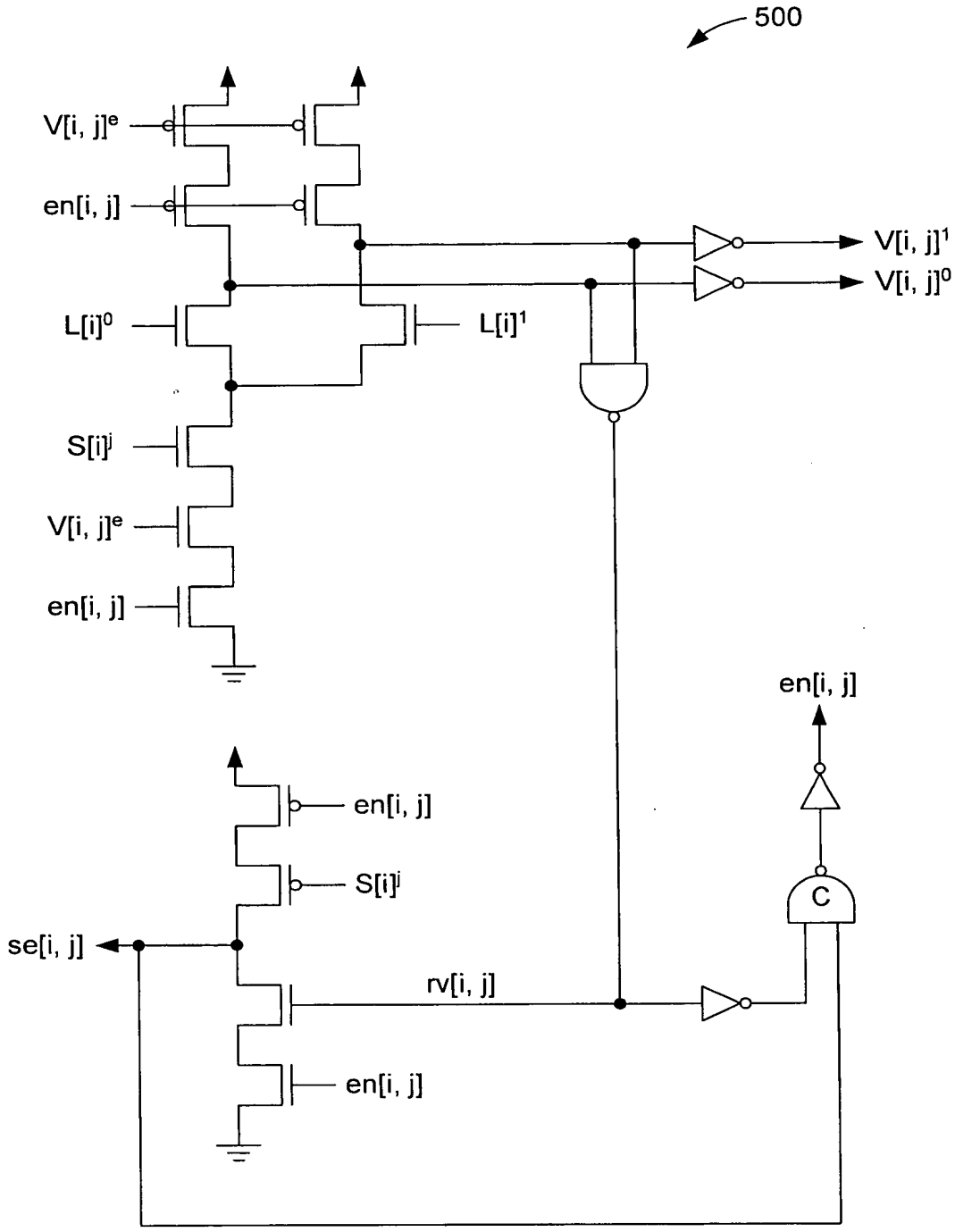

500
V[i, j]^e
en[i, j]
V[i, j]^1
V[i, j]^0
L[i]^0
L[i]^1
S[i]^j
V[i, j]^e
en[i, j]
en[i, j]
en[i, j]
S[i]^j
C
se[i, j]
rv[i, j]
en[i, j]

Fig. 5

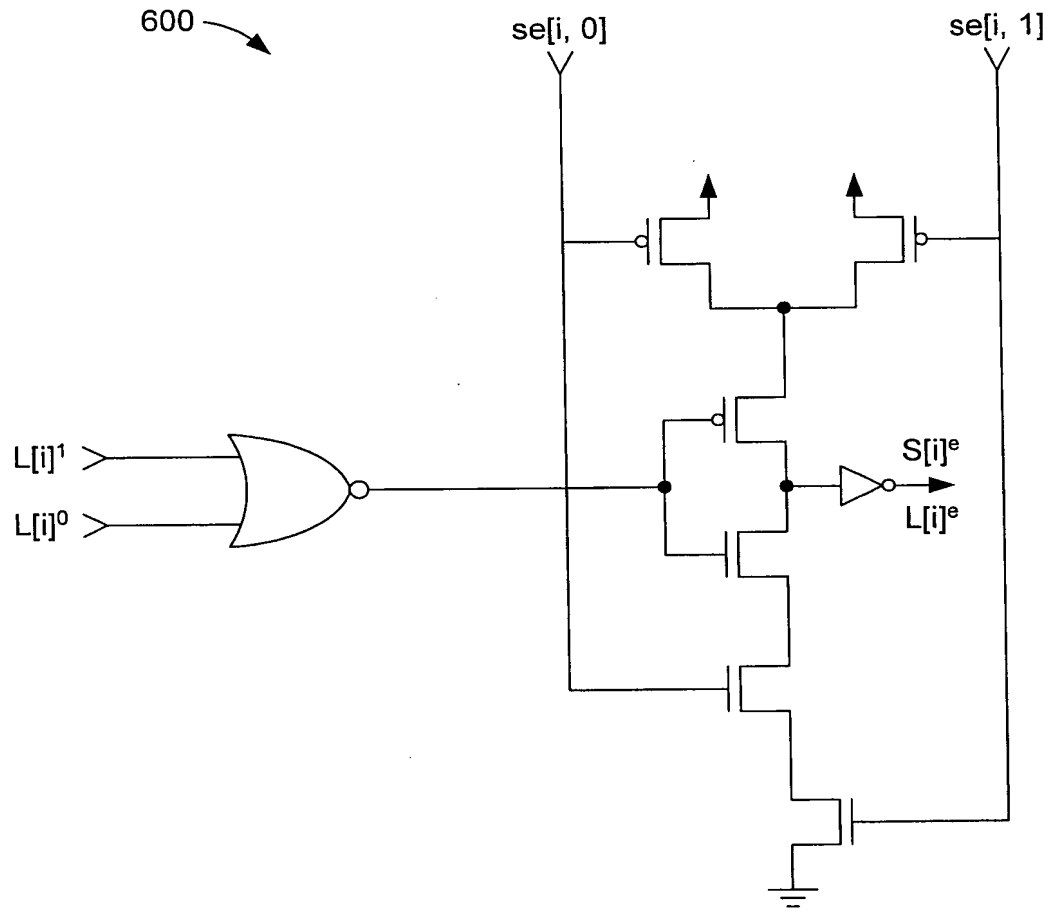


Fig. 6

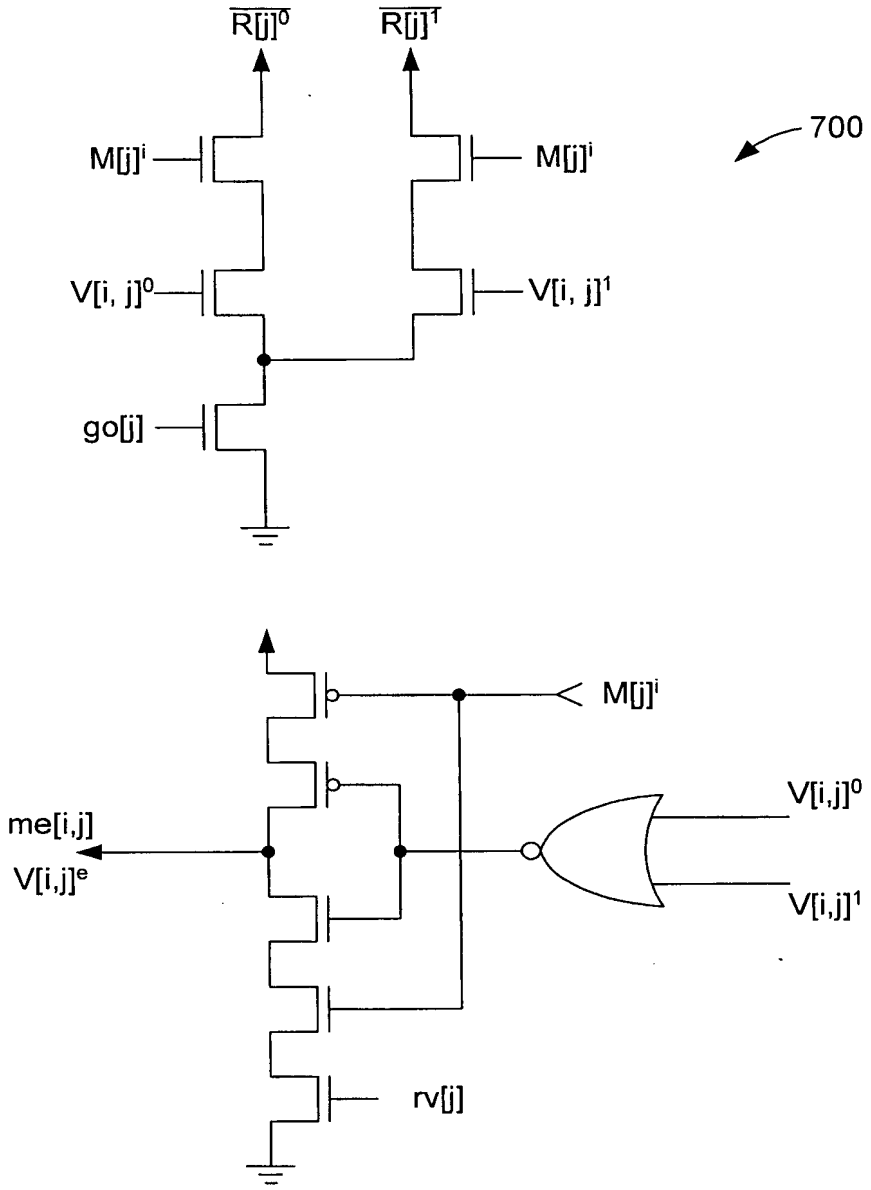

$\overline{R[j]^0}$
$\overline{R[j]^1}$
700
$M[j]^i$
$M[j]^i$
$V[i, j]^0$
$V[i, j]^1$
go[j]
$M[j]^i$
me[i,j]
$V[i,j]^e$
$V[i,j]^0$
$V[i,j]^1$
rv[j]

Fig. 7

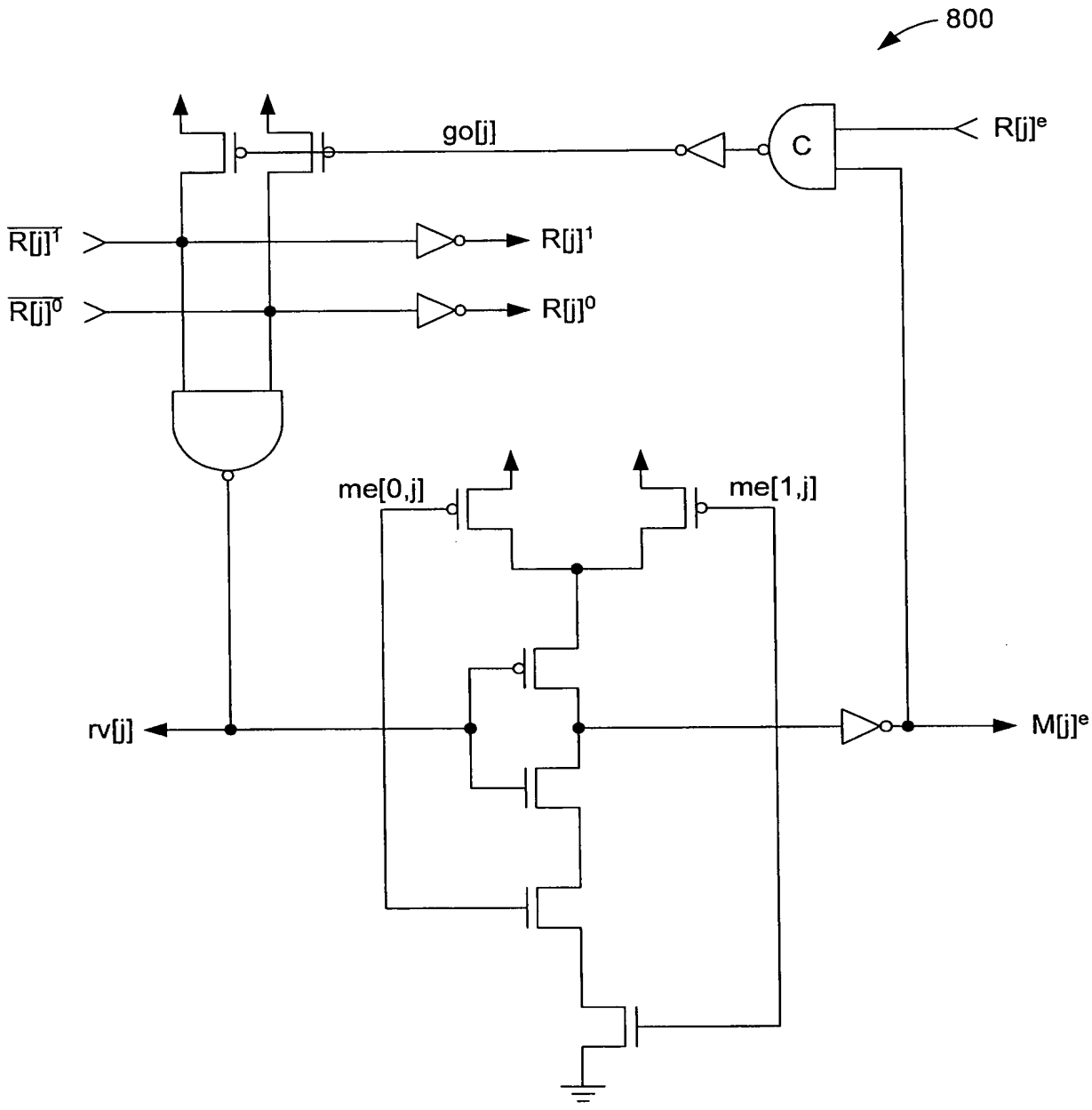

800
go[j]
C
R[j]^e
$\overline{R[j]^1}$
R[j]^1
$\overline{R[j]^0}$
R[j]^0
me[0,j]
me[1,j]
rv[j]
M[j]^e

Fig. 8

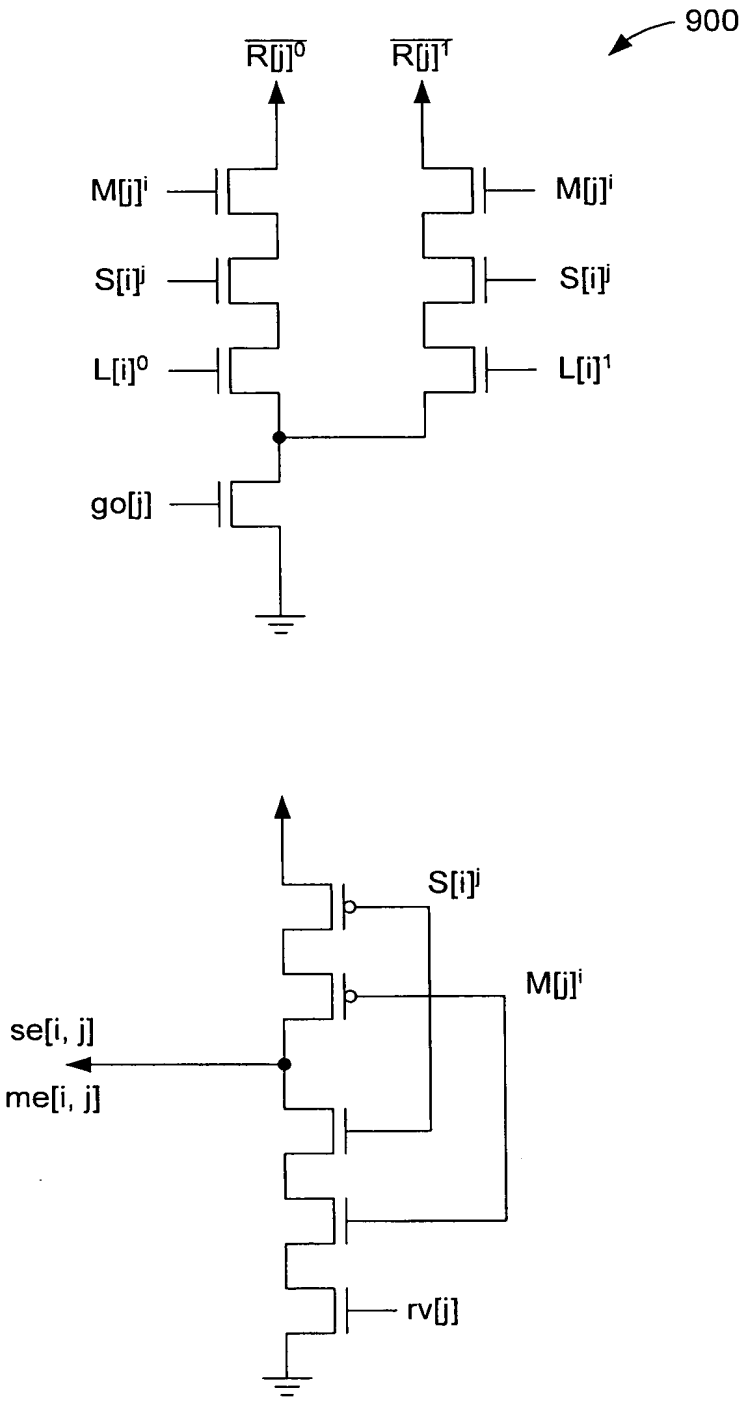

900
$\overline{R[j]^0}$
$\overline{R[j]^1}$
$M[j]^i$
$S[i]^j$
$L[i]^0$
$L[i]^1$
go[j]
$S[i]^j$
$M[j]^i$
se[i, j]
me[i, j]
rv[j]

Fig. 9

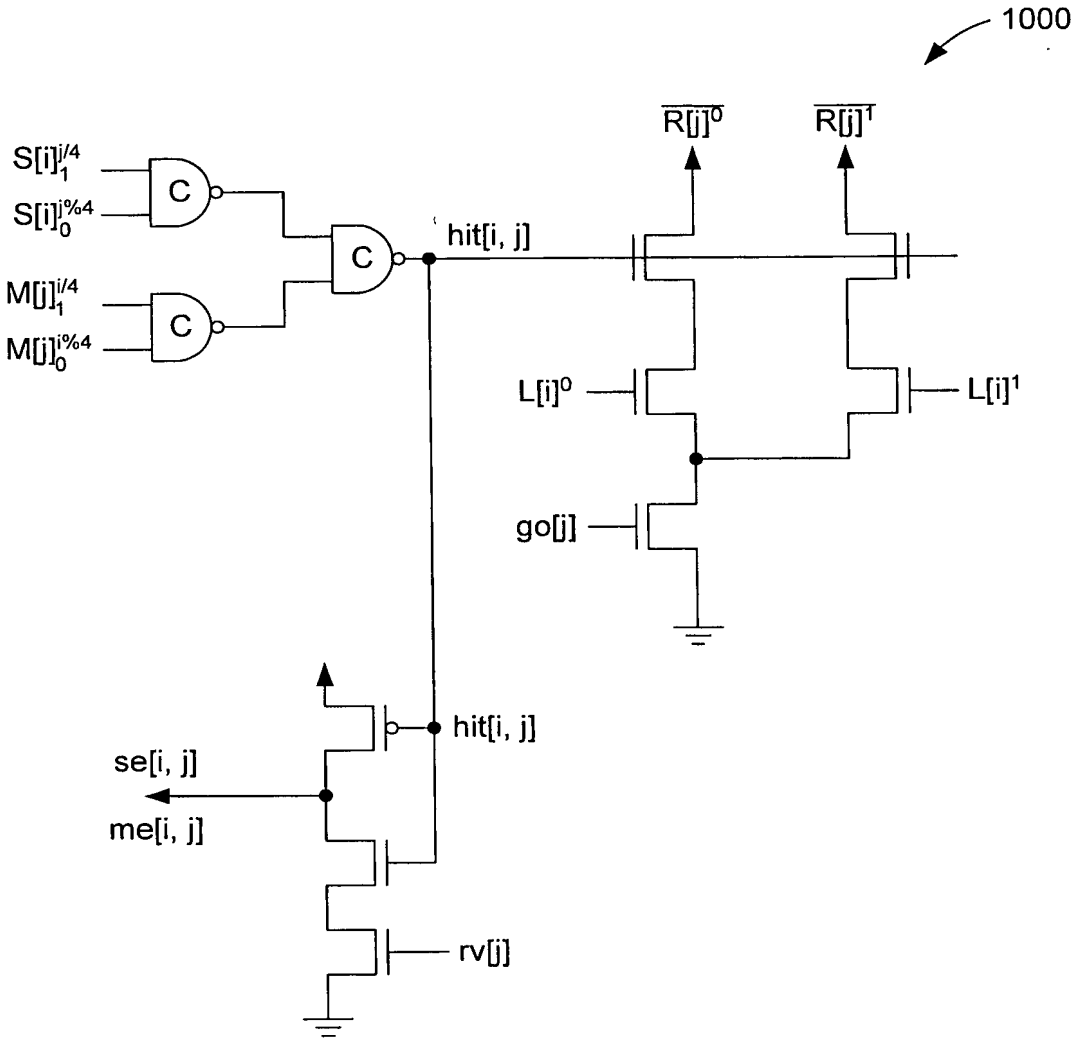

1000
$\overline{R[j]^0}$
$\overline{R[j]^1}$
$S[i]_1^{j/4}$
$S[i]_0^{j\%4}$
C
C
hit[i, j]
$M[j]_1^{i/4}$
$M[j]_0^{i\%4}$
C
$L[i]^0$
$L[i]^1$
go[j]
hit[i, j]
se[i, j]
me[i, j]
rv[j]

Fig. 10

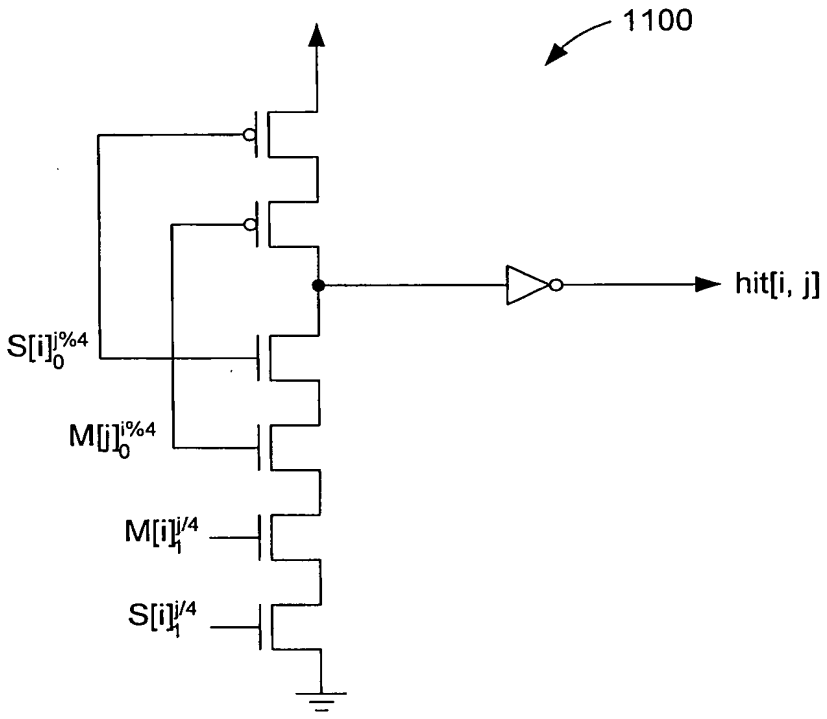


Fig. 11

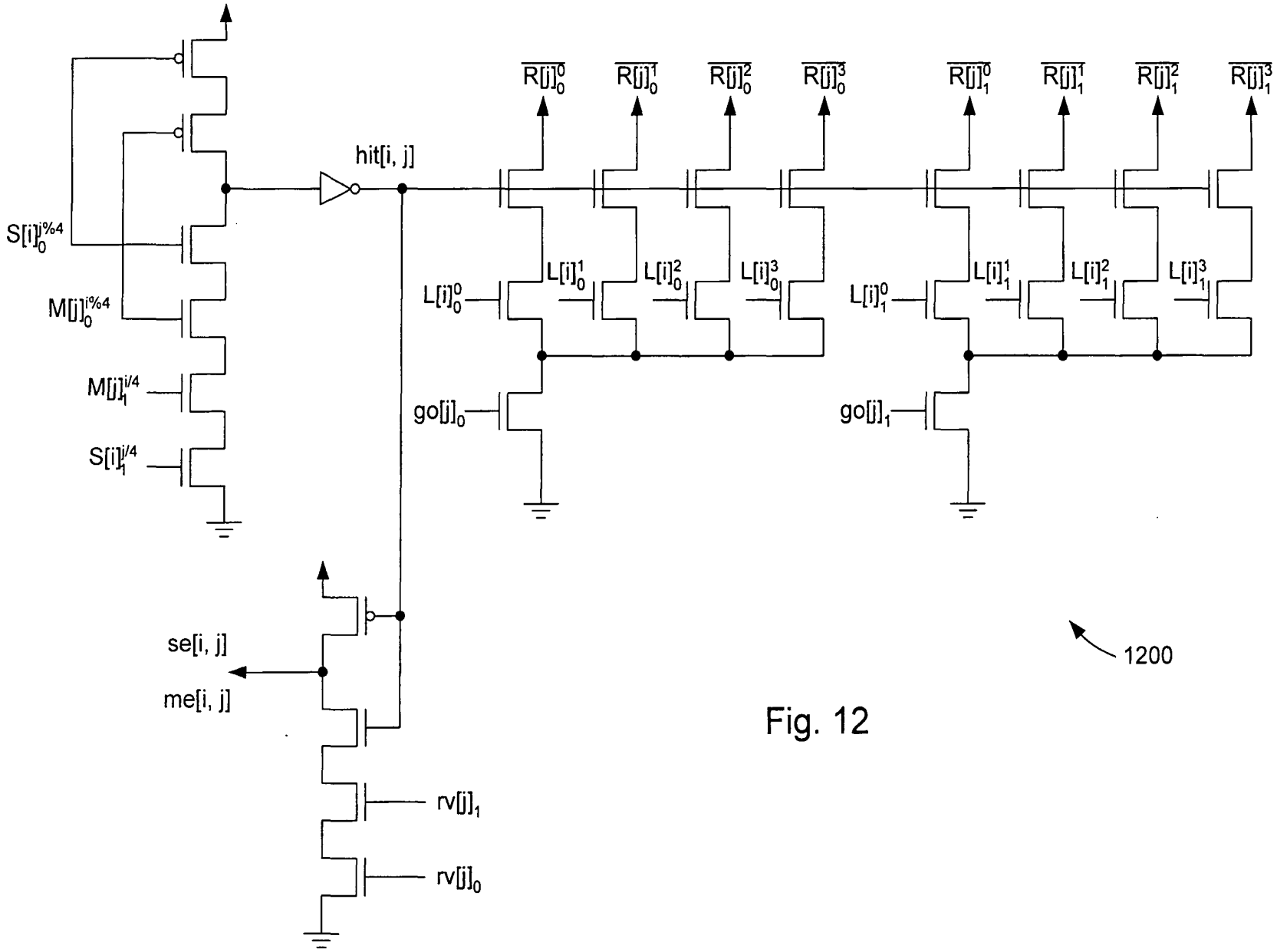

hit[i, j]
$S[i]_0^{j\%4}$
$M[j]_0^{i\%4}$
$M[j]_1^{j/4}$
$S[i]_1^{j/4}$
$\overline{R[j]_0^0}$
$\overline{R[j]_0^1}$
$\overline{R[j]_0^2}$
$\overline{R[j]_0^3}$
$\overline{R[j]_1^0}$
$\overline{R[j]_1^1}$
$\overline{R[j]_1^2}$
$\overline{R[j]_1^3}$
$L[i]_0^0$
$L[i]_0^1$
$L[i]_0^2$
$L[i]_0^3$
$L[i]_1^0$
$L[i]_1^1$
$L[i]_1^2$
$L[i]_1^3$
$go[j]_0$
$go[j]_1$
se[i, j]
me[i, j]
$rv[j]_1$
$rv[j]_0$
1200

Fig. 12

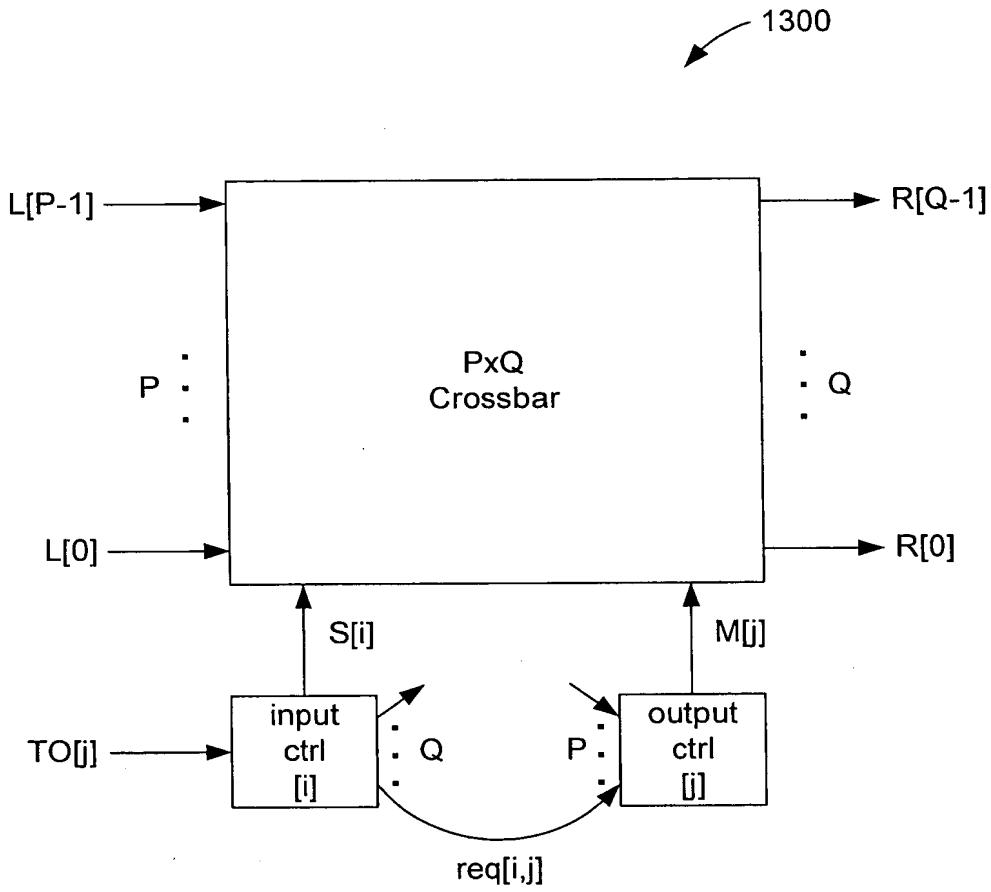

1300
L[P-1]
R[Q-1]
P
PxQ
Crossbar
Q
L[0]
R[0]
S[i]
M[j]
TO[j]
input
ctrl
[i]
Q
P
output
ctrl
[j]
req[i,j]

Fig. 13

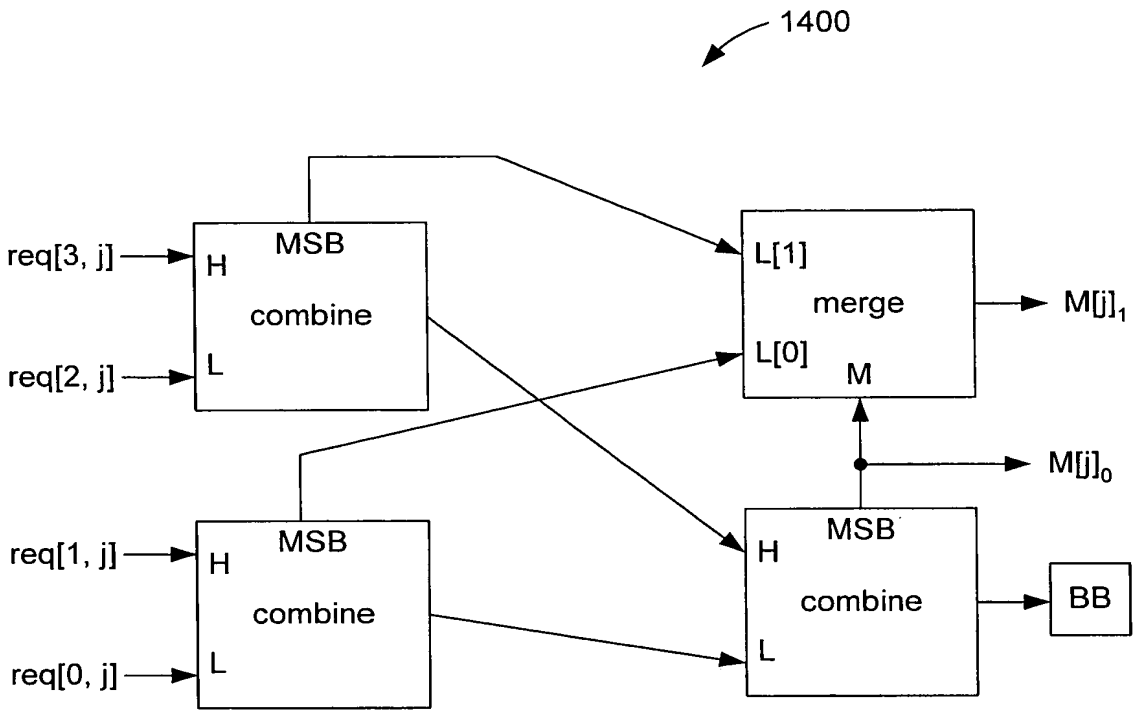

1400
req[3, j]
req[2, j]
MSB
H
combine
L
L[1]
merge
L[0]
M
M[j]1
M[j]0
req[1, j]
req[0, j]
MSB
H
combine
L
MSB
H
combine
L
BB

Fig. 14

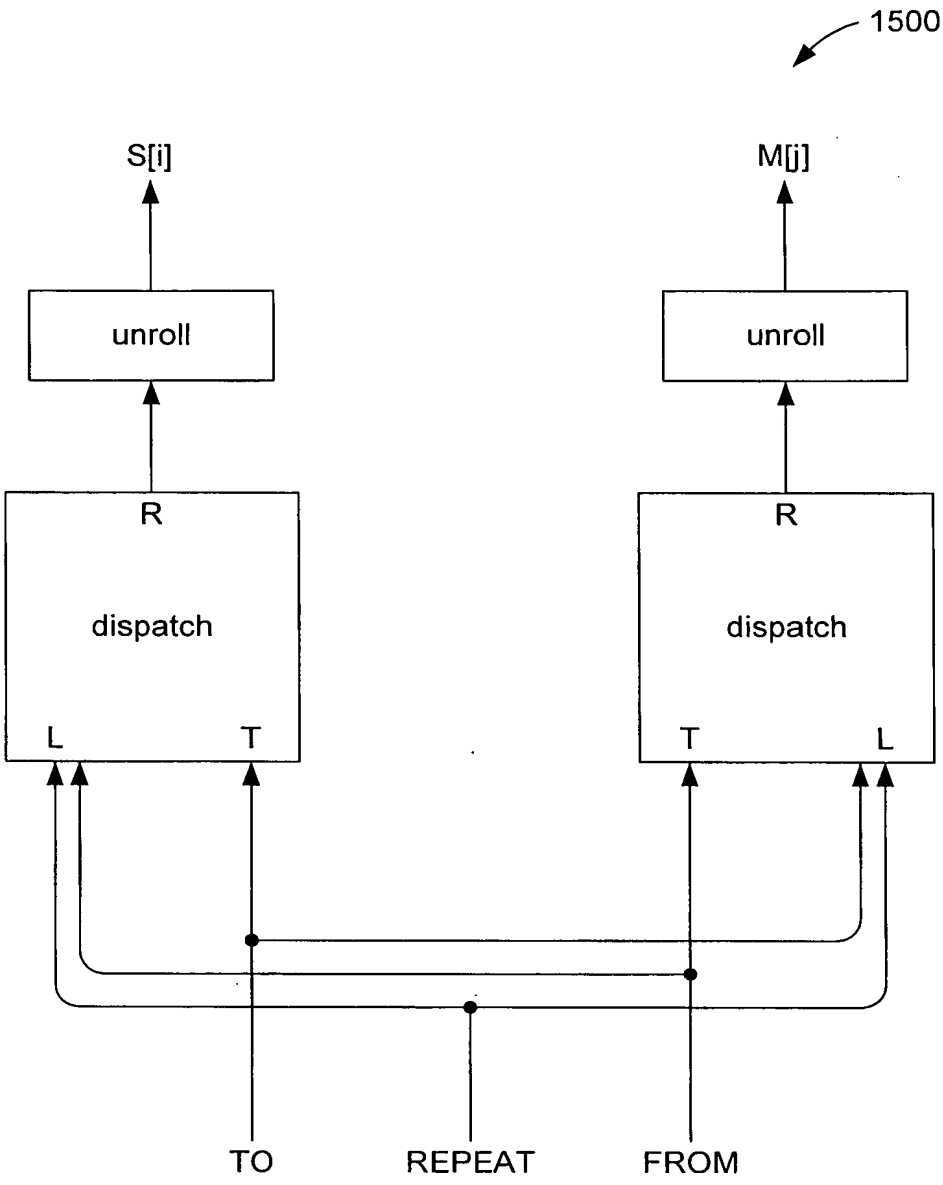

1500
S[i]
M[j]
unroll
unroll
R
R
dispatch
dispatch
L
T
T
L
TO
REPEAT
FROM

Fig. 15

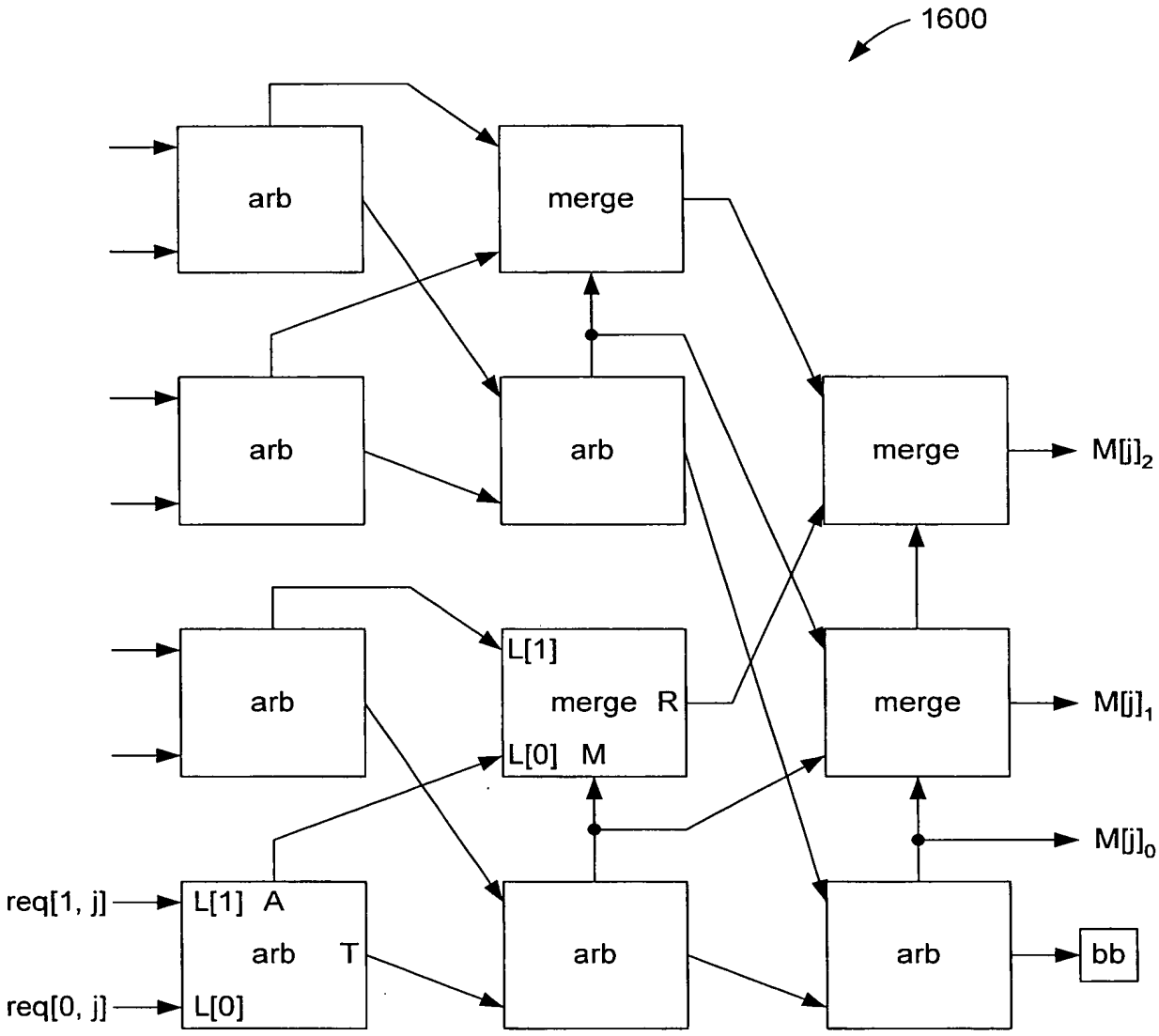

1600
arb
merge
arb
arb
merge
M[j]2
arb
L[1]
merge R
L[0] M
merge
M[j]1
M[j]0
req[1, j]
L[1] A
arb T
req[0, j]
L[0]
arb
arb
bb

Fig. 16

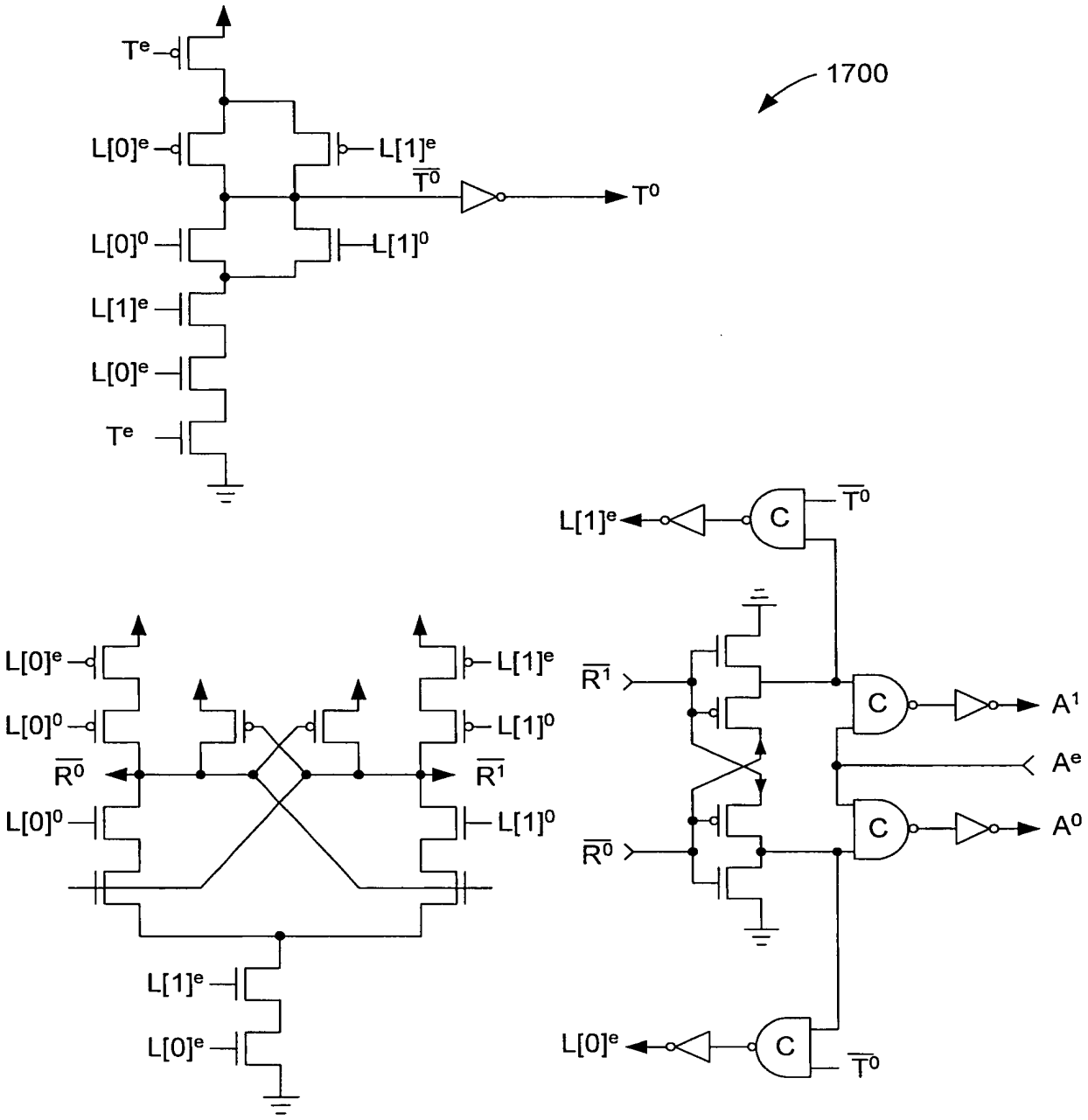

1700
T^e
$L[0]^e$
$L[1]^e$
$\overline{T^0}$
T^0
$L[0]^0$
$L[1]^0$
$L[1]^e$
$L[0]^e$
T^e
$L[1]^e$
$\overline{T^0}$
C
$\overline{R^1}$
C
A^1
A^e
$\overline{R^0}$
C
A^0
$L[0]^e$
C
$\overline{T^0}$
$L[0]^e$
$L[0]^0$
$\overline{R^0}$
$L[0]^0$
$L[1]^e$
$L[1]^0$
$\overline{R^1}$
$L[1]^0$
$L[1]^e$
$L[0]^e$

Fig. 17

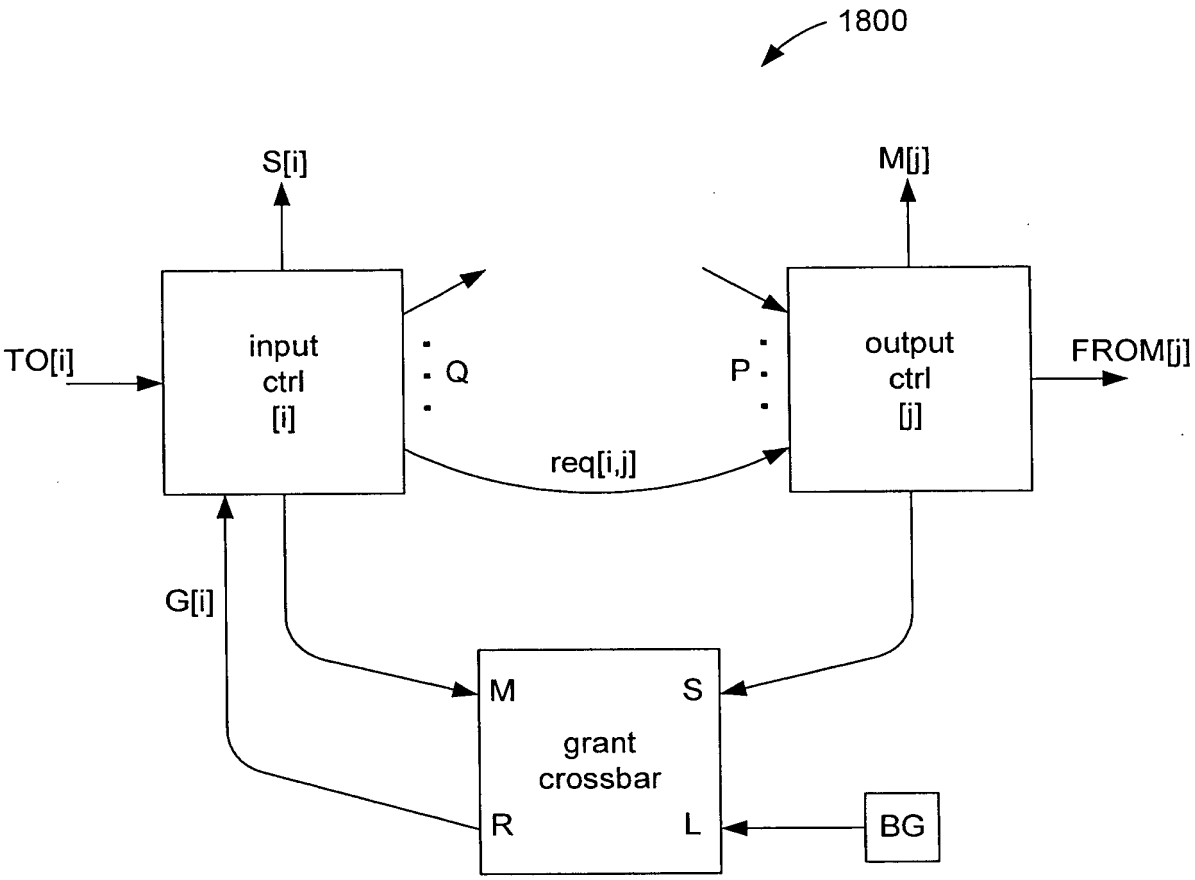

1800
S[i]
M[j]
TO[i]
input
ctrl
[i]
Q
P
output
ctrl
[j]
FROM[j]
req[i,j]
G[i]
M
S
grant
crossbar
R
L
BG

Fig. 18

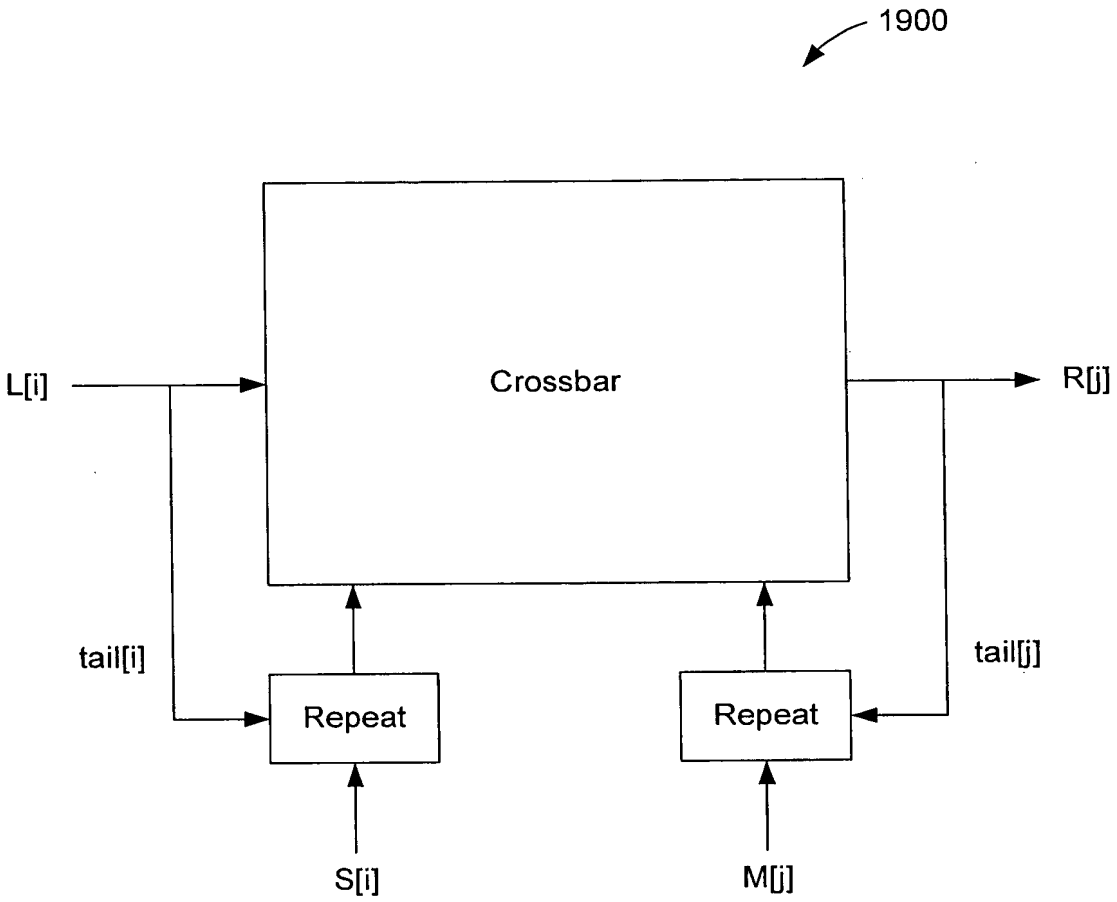


Fig. 19

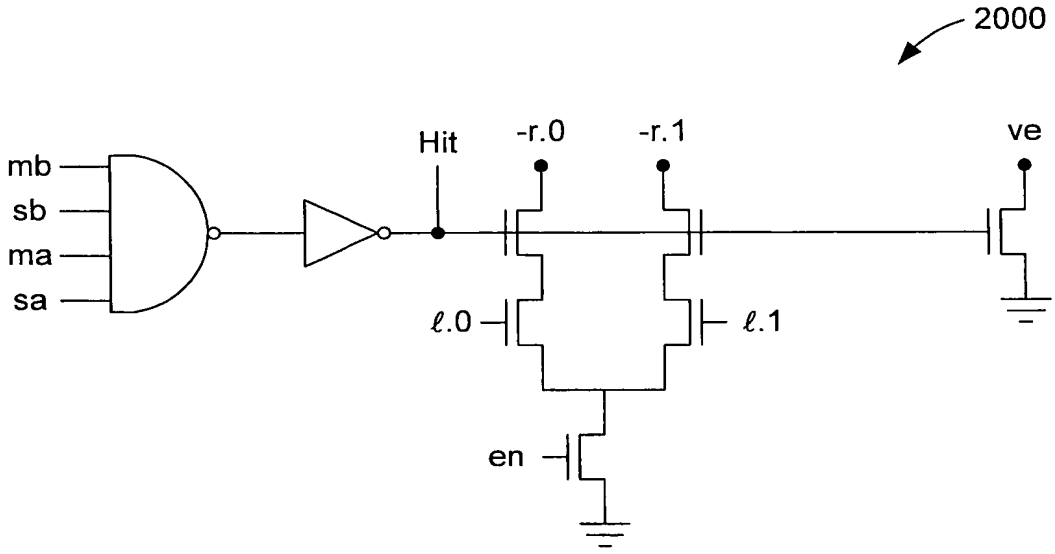


Fig. 20A

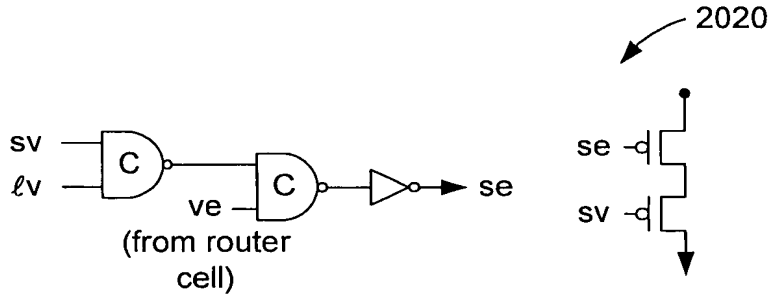


Fig. 20B

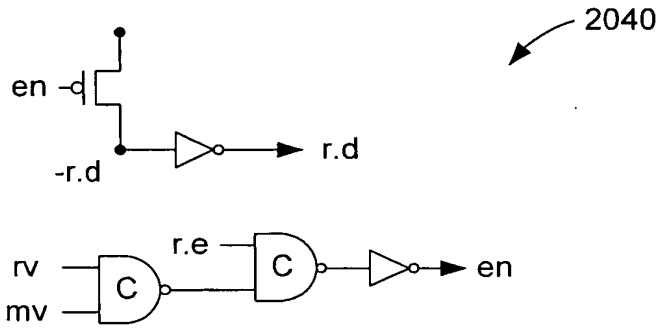


Fig. 20C

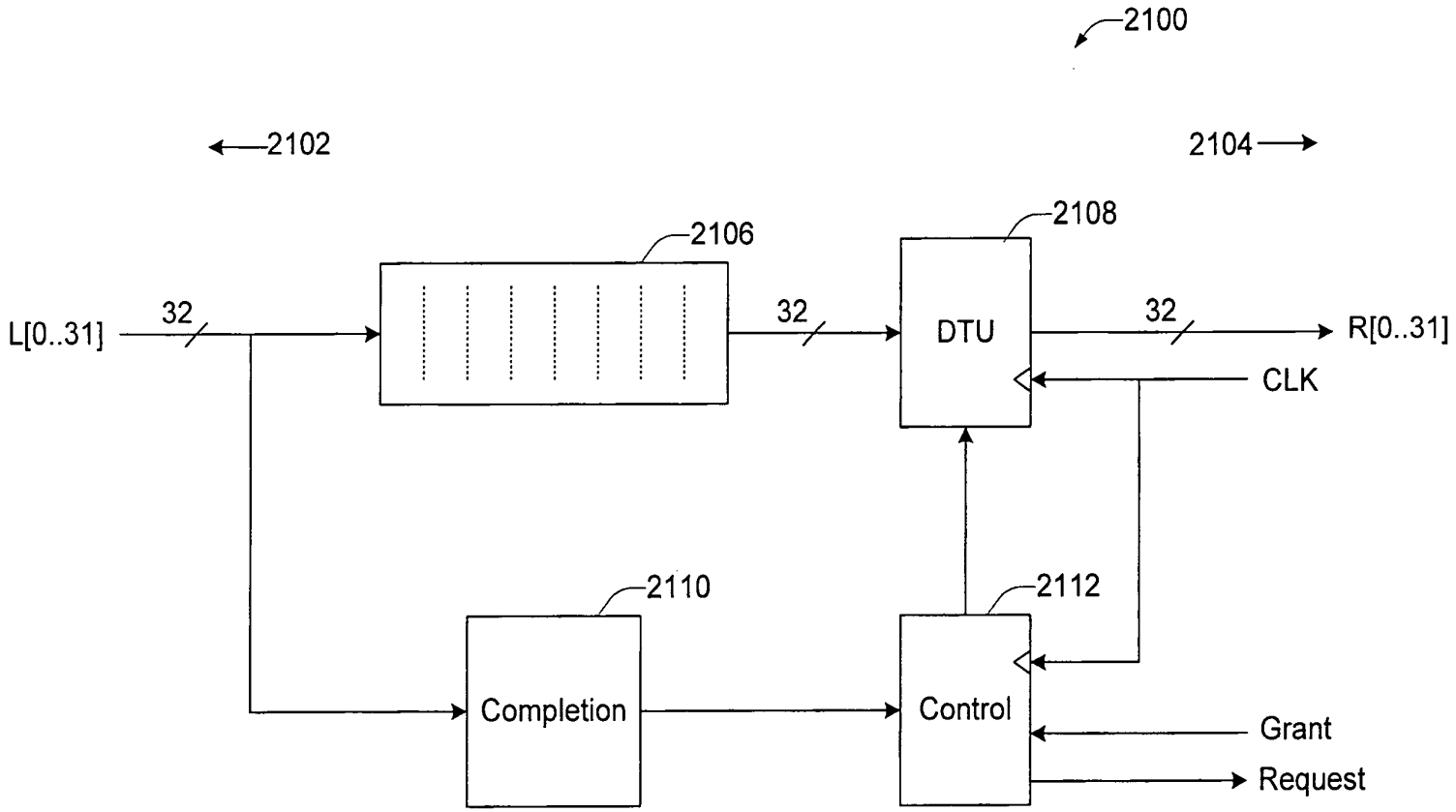

2100
2102
2104
2106
2108
L[0..31]
32
32
32
DTU
R[0..31]
CLK
2110
2112
Completion
Control
Grant
Request

FIG. 21

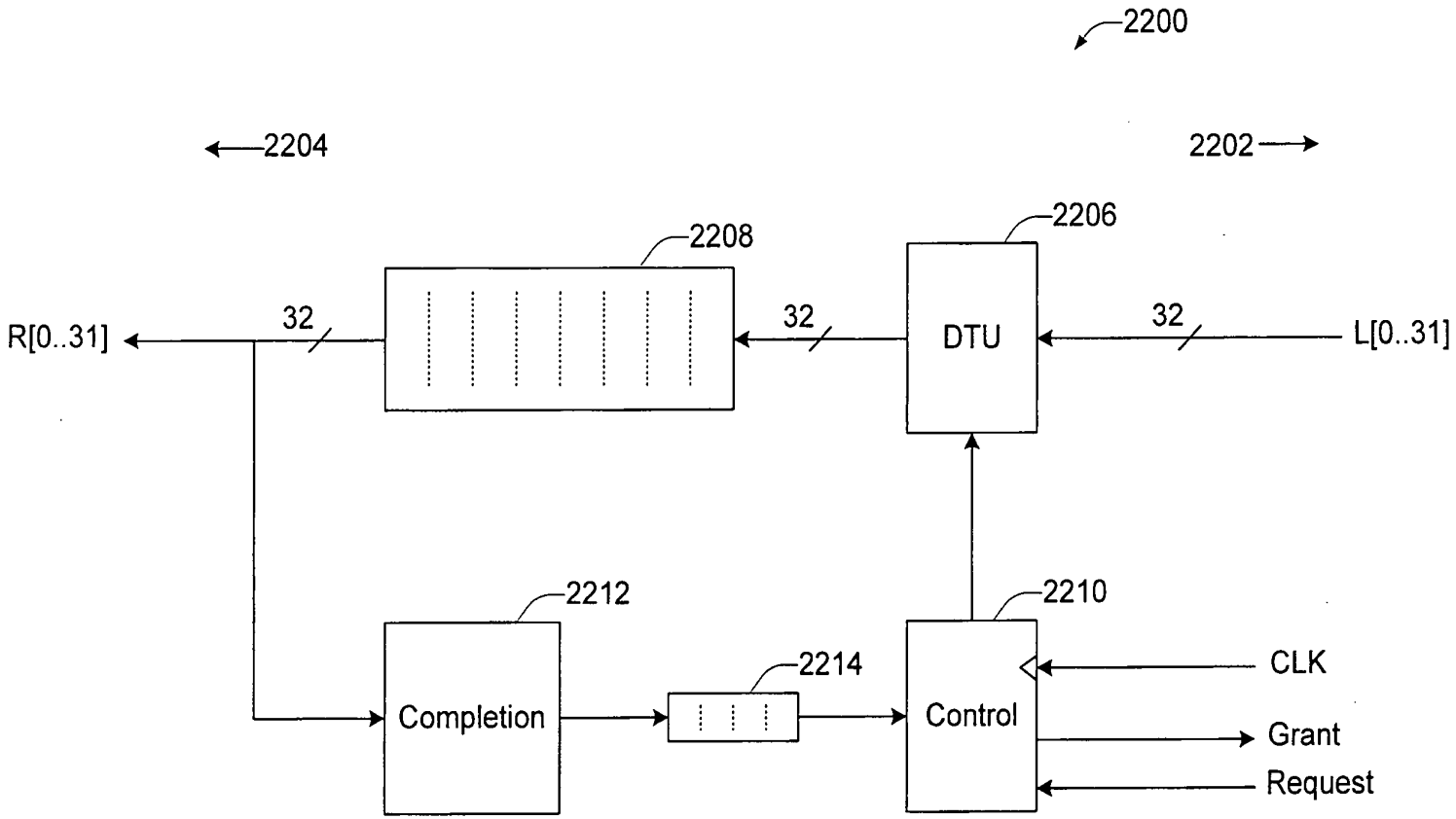

2200
2204
2202
2208
2206
R[0..31]
32
32
32
DTU
L[0..31]
2212
2210
2214
CLK
Completion
Control
Grant
Request

FIG. 22

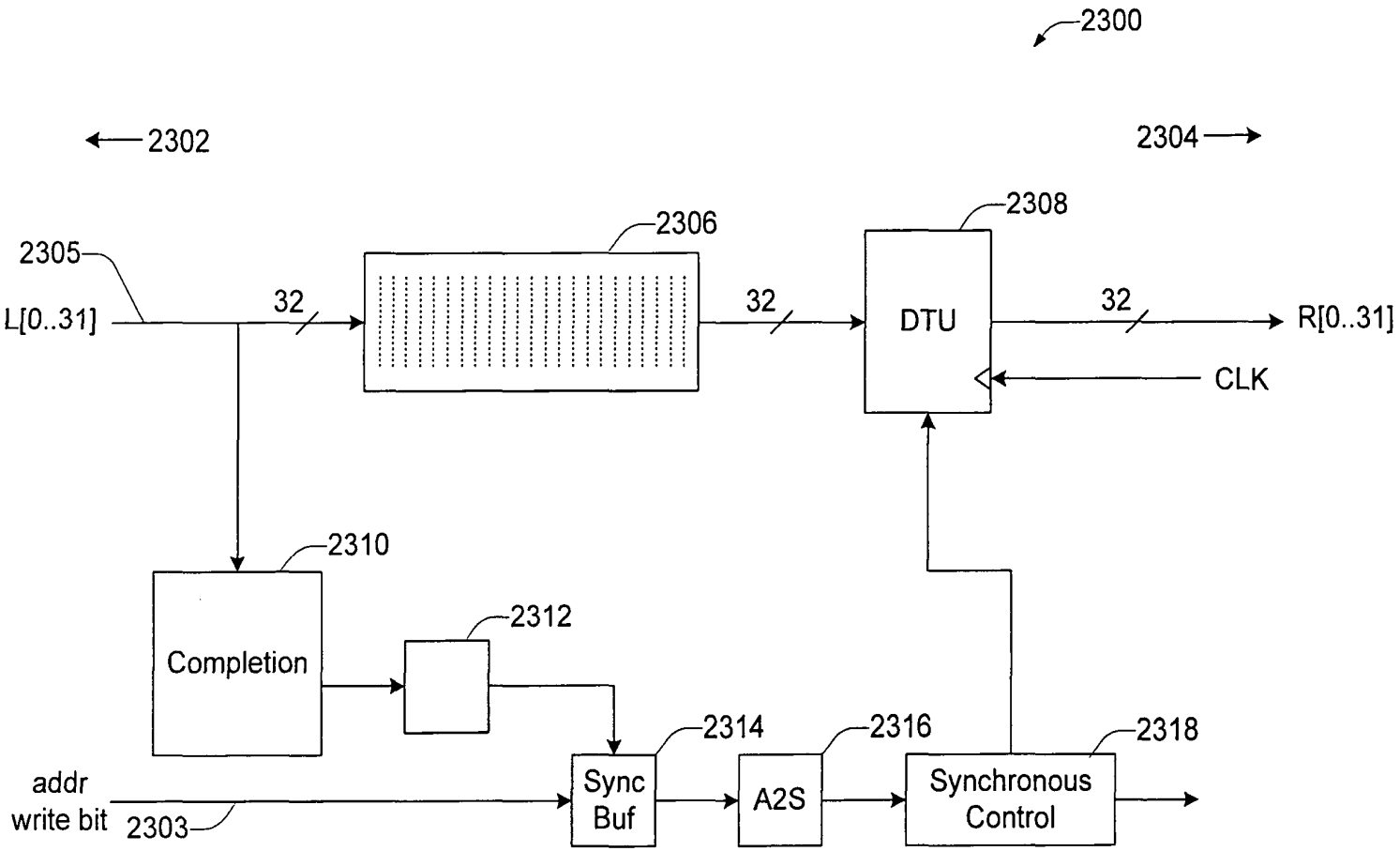

2300
2302
2304
2305
L[0..31]
32
2306
32
2308
DTU
32
R[0..31]
CLK
2310
Completion
2312
2314
Sync Buf
2316
A2S
2318
Synchronous Control
addr
write bit
2303

FIG. 23

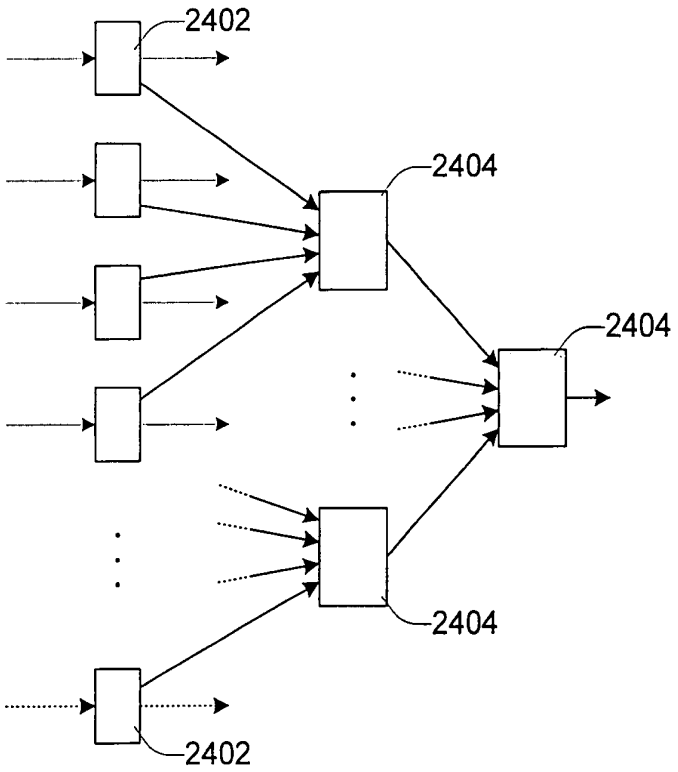


FIG. 24

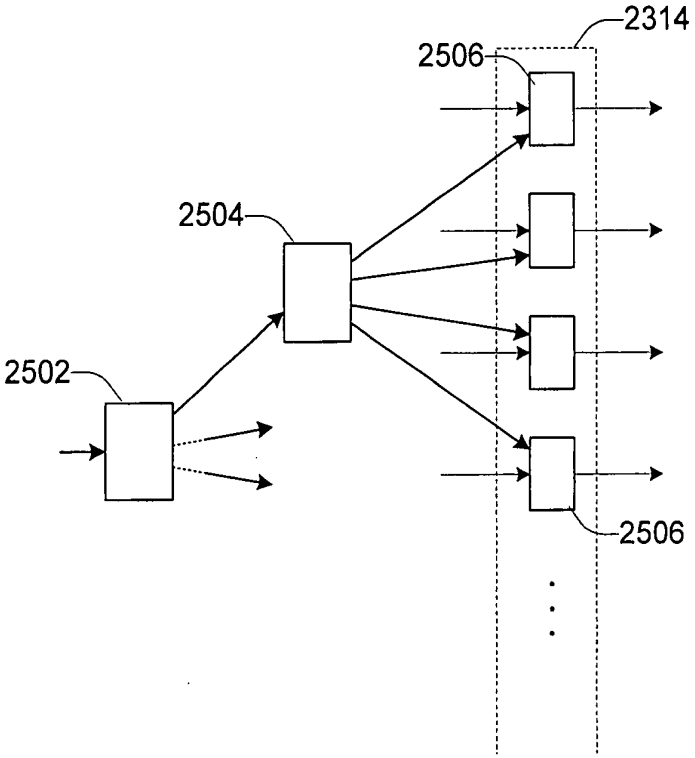


FIG. 25

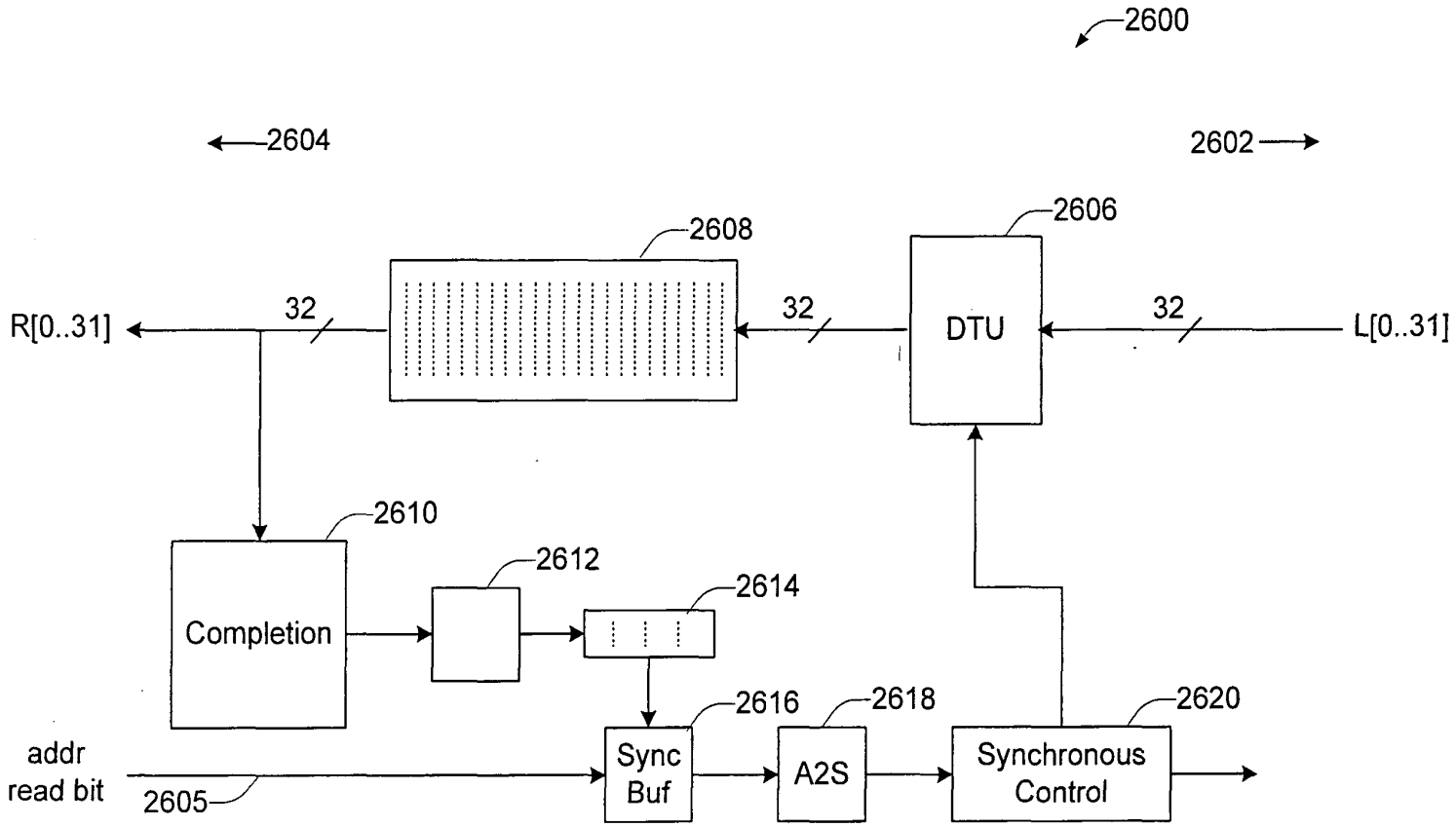

2600
2604
2602
2608
2606
R[0..31]
32
32
DTU
32
L[0..31]
2610
2612
2614
Completion
2616
2618
2620
addr
read bit
2605
Sync
Buf
A2S
Synchronous
Control

FIG. 26

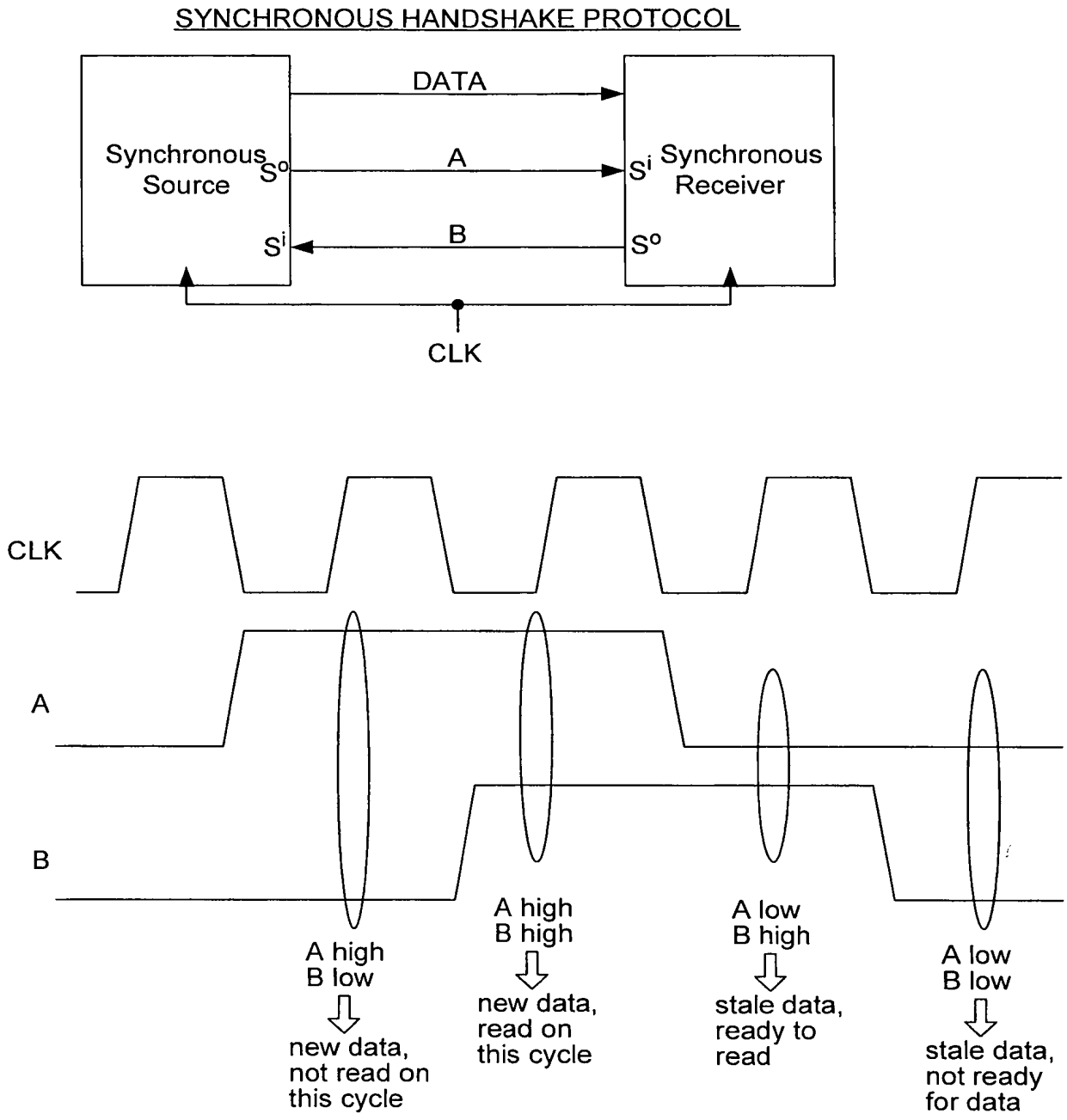

SYNCHRONOUS HANDSHAKE PROTOCOL
DATA
Synchronous Source
So
A
Si
Synchronous Receiver
Si
B
So
CLK
CLK
A
B
A high
B low
new data,
not read on
this cycle
A high
B high
new data,
read on
this cycle
A low
B high
stale data,
ready to
read
A low
B low
stale data,
not ready
for data

FIG. 27

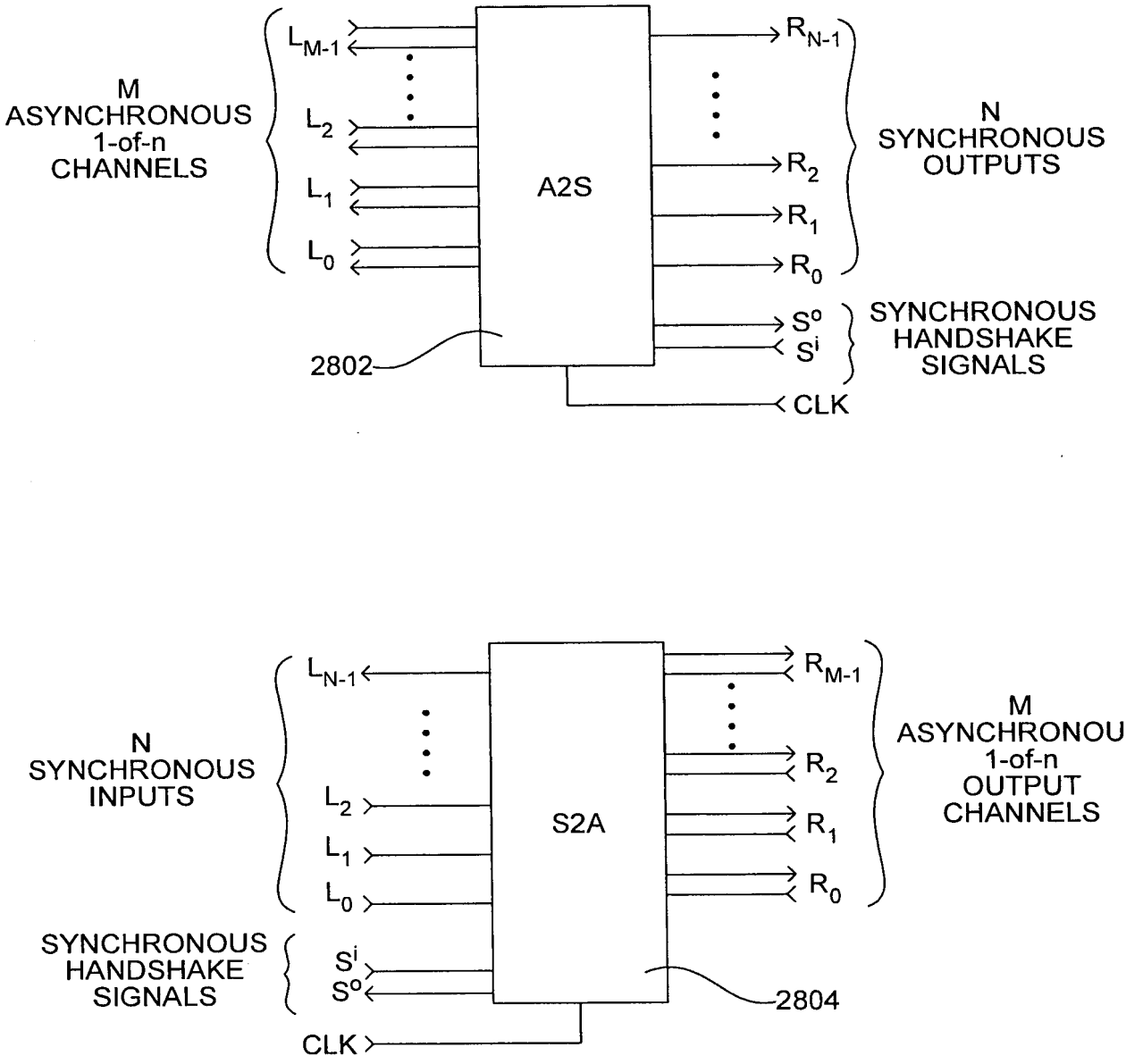

M
ASYNCHRONOUS
1-of-n
CHANNELS
L_{M-1}
L_2
L_1
L_0
A2S
R_{N-1}
R_2
R_1
R_0
N
SYNCHRONOUS
OUTPUTS
S^o
S^i
SYNCHRONOUS
HANDSHAKE
SIGNALS
2802
CLK
N
SYNCHRONOUS
INPUTS
L_{N-1}
L_2
L_1
L_0
S2A
R_{M-1}
R_2
R_1
R_0
M
ASYNCHRONOUS
1-of-n
OUTPUT
CHANNELS
SYNCHRONOUS
HANDSHAKE
SIGNALS
S^i
S^o
CLK
2804

FIG. 28

TOP-LEVEL DECOMPOSITION

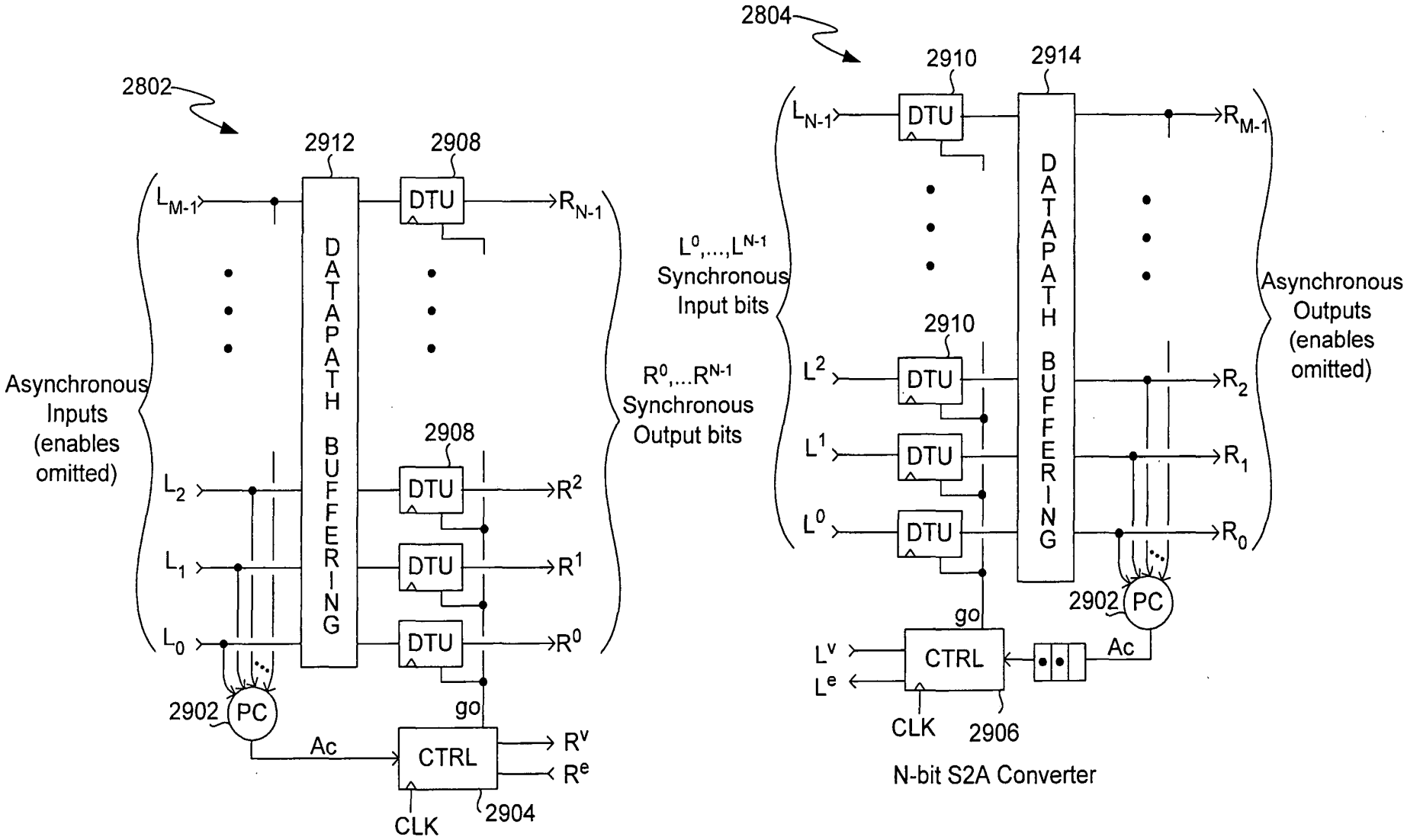


FIG. 29

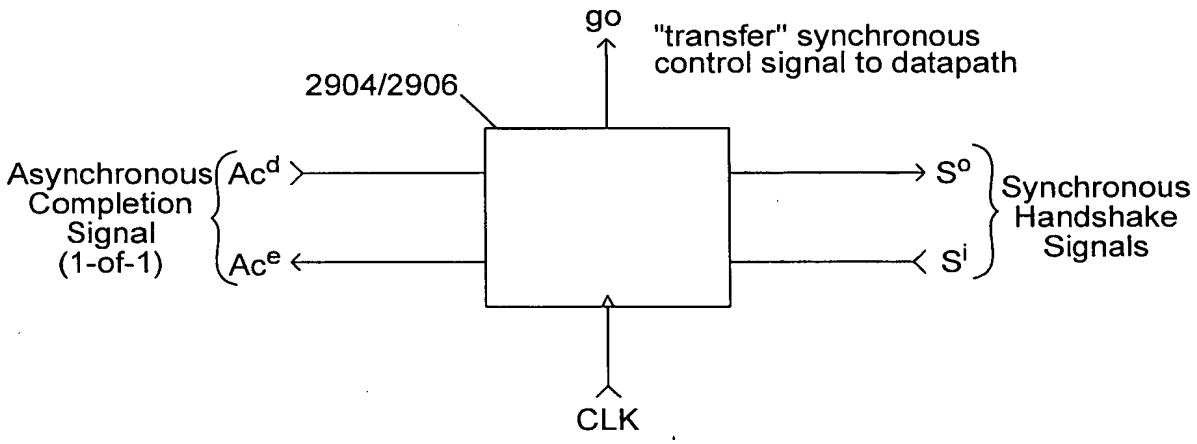
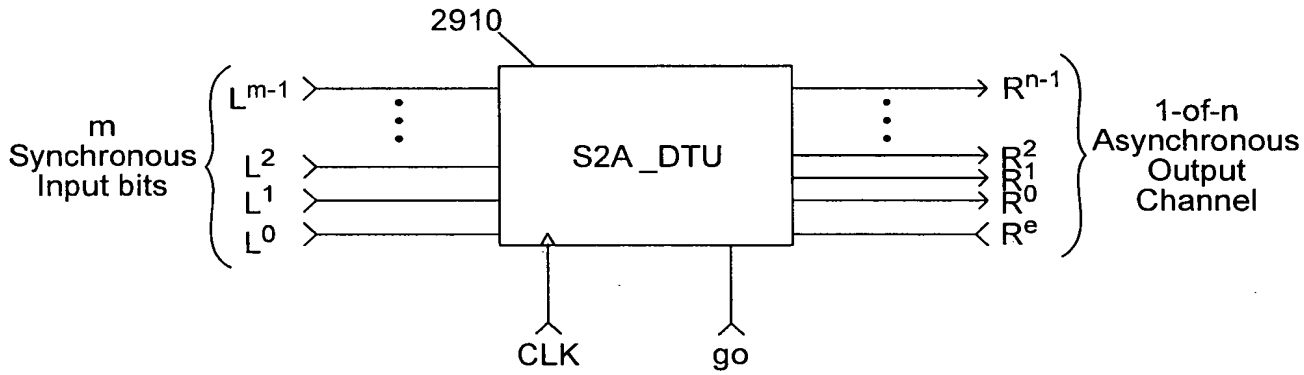

go
"transfer" synchronous control signal to datapath
2904/2906
Asynchronous Completion Signal (1-of-1)
Ac^d
Ac^e
S^o
S^i
Synchronous Handshake Signals
CLK

FIG. 30


2910
m Synchronous Input bits
L^{m-1}
L^2
L^1
L^0
S2A_DTU
R^{n-1}
R^2
R^1
R^0
R^e
1-of-n Asynchronous Output Channel
CLK
go

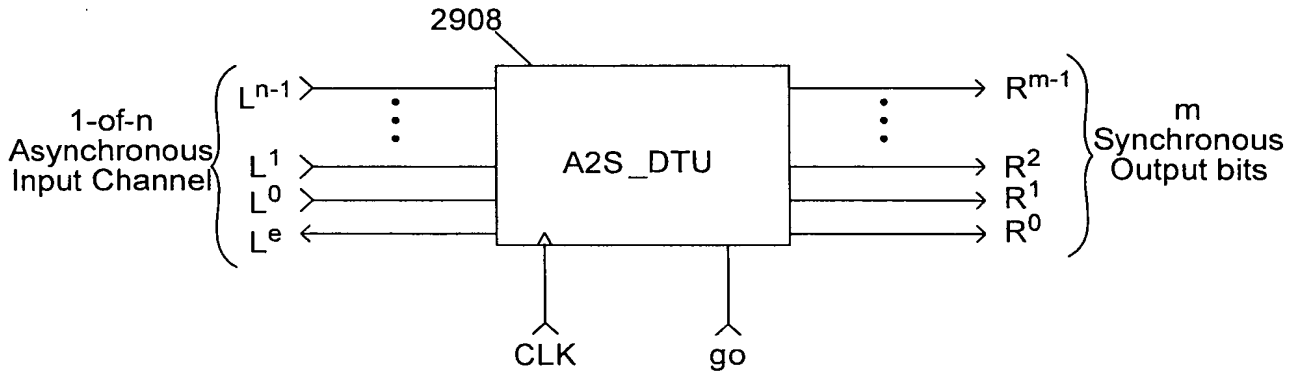
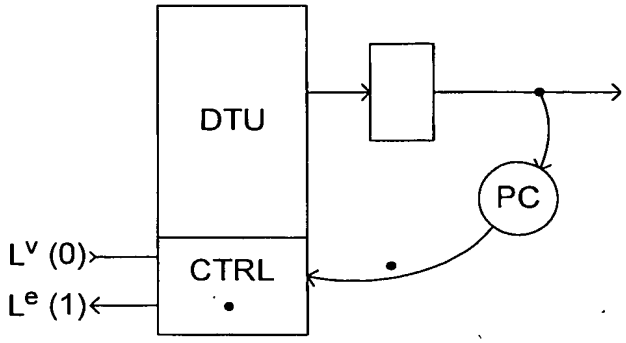
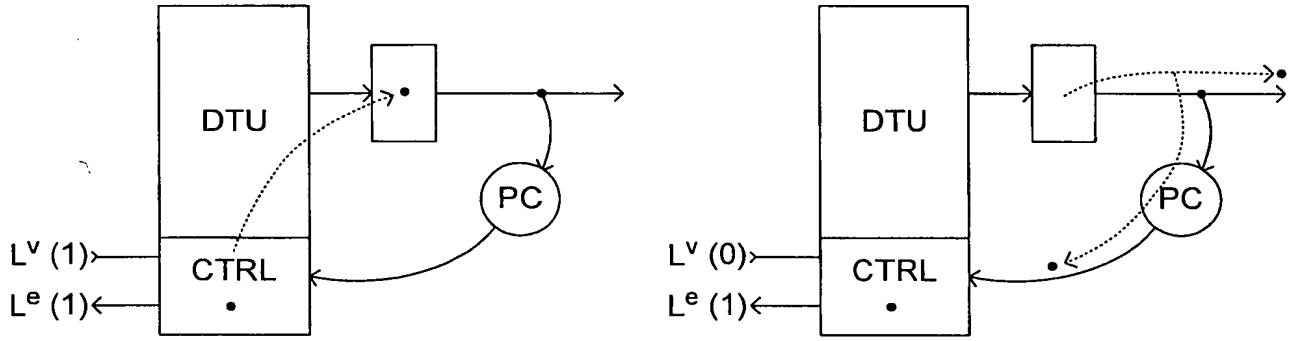

2908
1-of-n Asynchronous Input Channel
L^{n-1}
L^1
L^0
L^e
A2S_DTU
R^{m-1}
R^2
R^1
R^0
m Synchronous Output bits
CLK
go

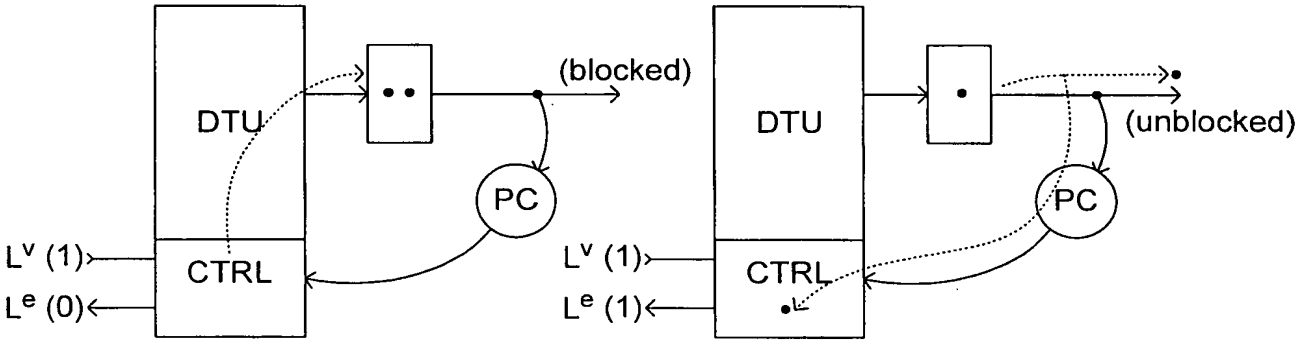
FIG. 31



(A) Reset Condition

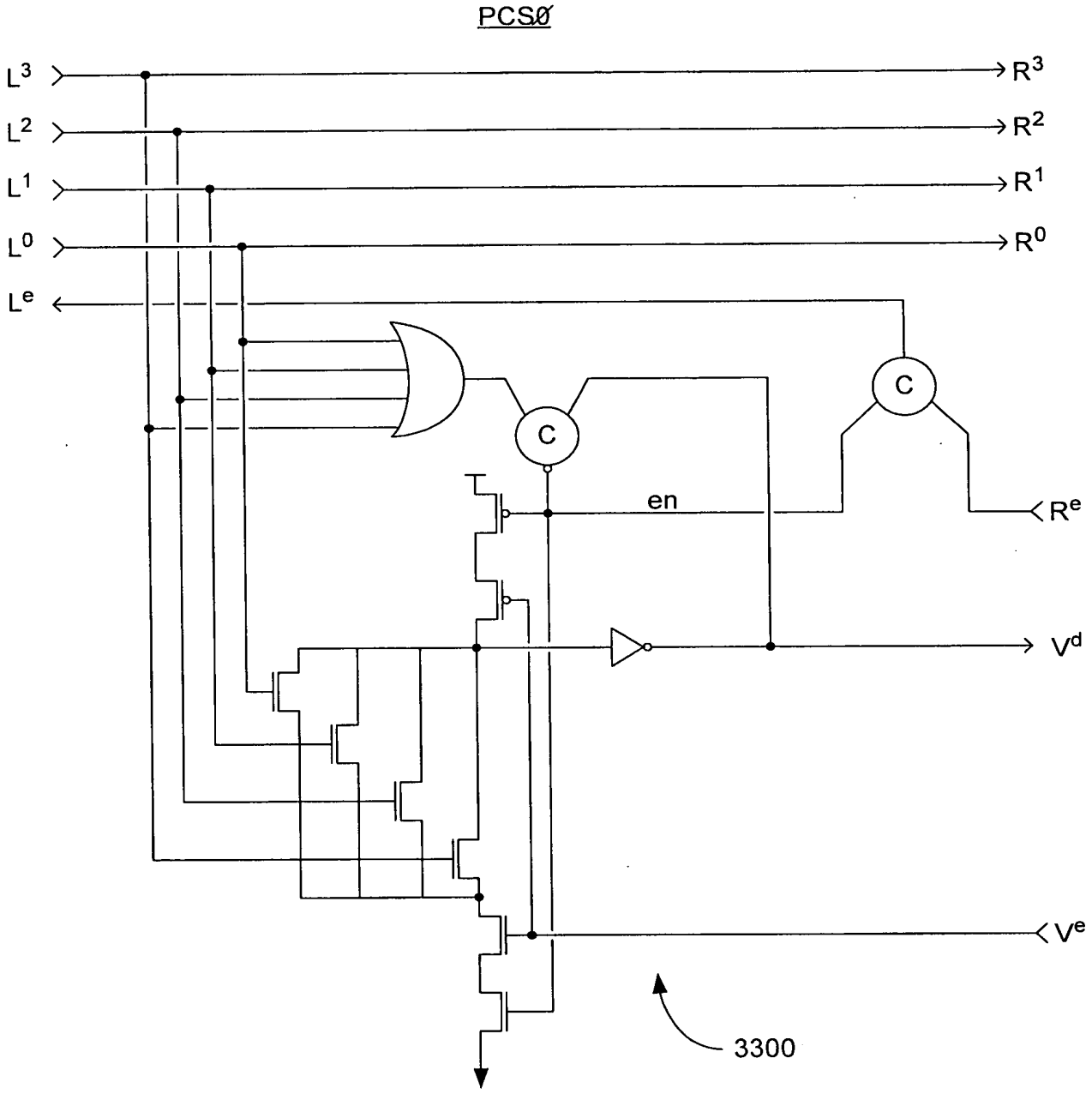


(B) Normal Operation



(C) Stall Condition
(Asynchronous Side Stalls)

FIG. 32



* [L?x; R!x, V!]

FIG. 33

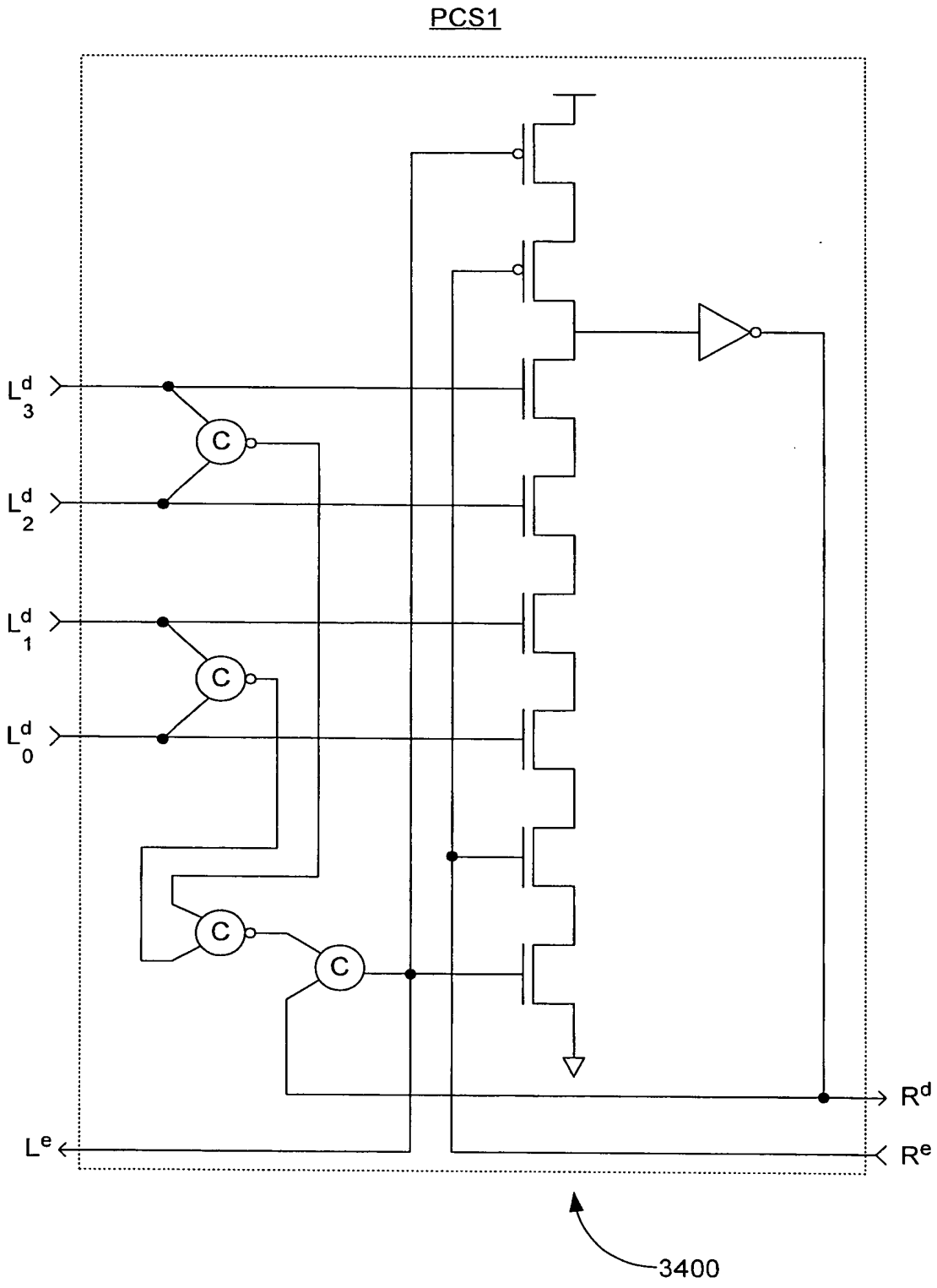
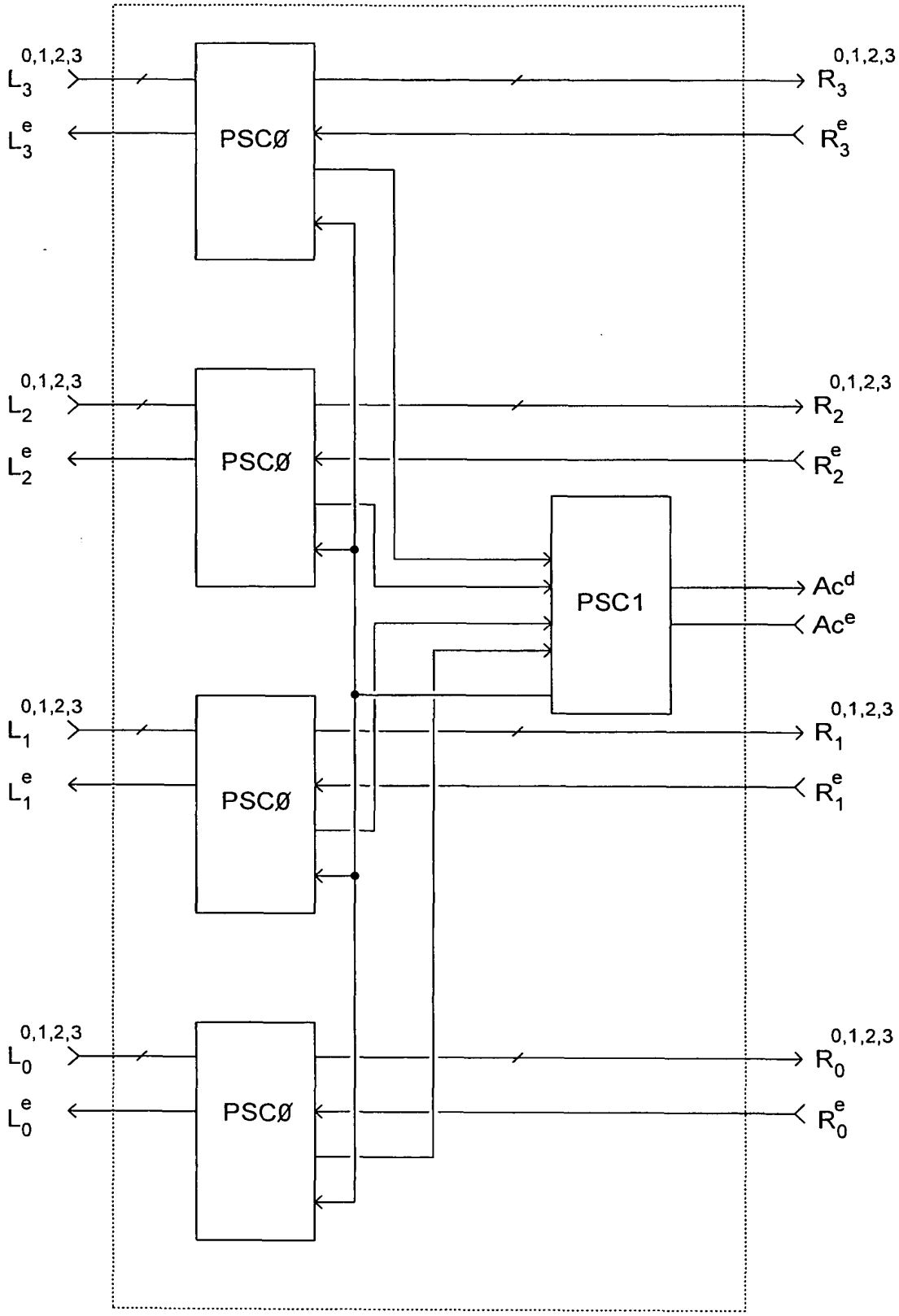


FIG. 34



8-bit 1-of-4 PC Tree

FIG. 35

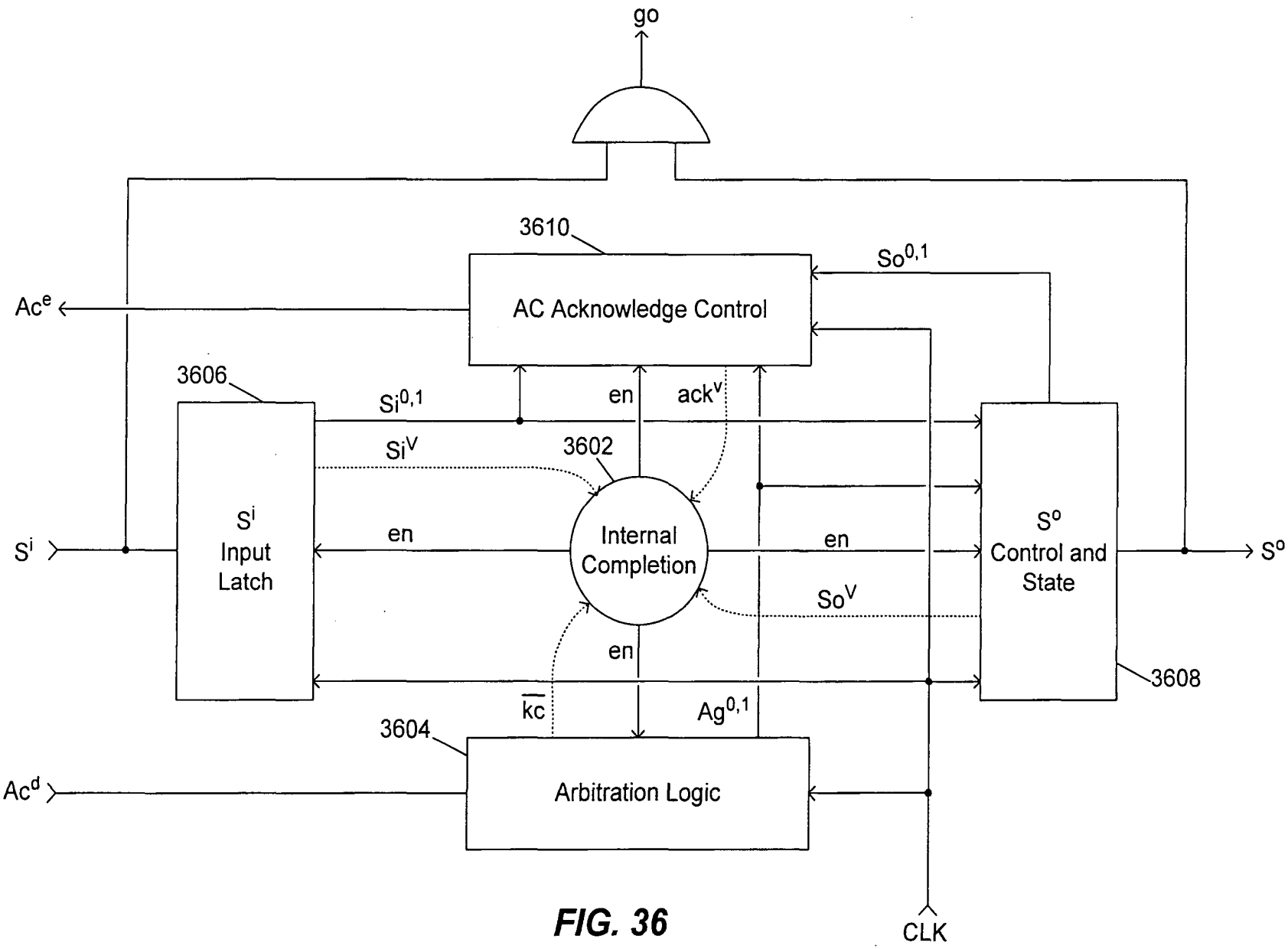

go
3610
AC Acknowledge Control
$So^{0,1}$
Ac^e
3606
$Si^{0,1}$
en
ack^v
Si^v
3602
S^i
Input
Latch
en
Internal
Completion
en
S^o
Control and
State
S^o
S^i
So^v
en
3608
$\overline{kc}$
$Ag^{0,1}$
3604
Ac^d
Arbitration Logic
CLK

FIG. 36

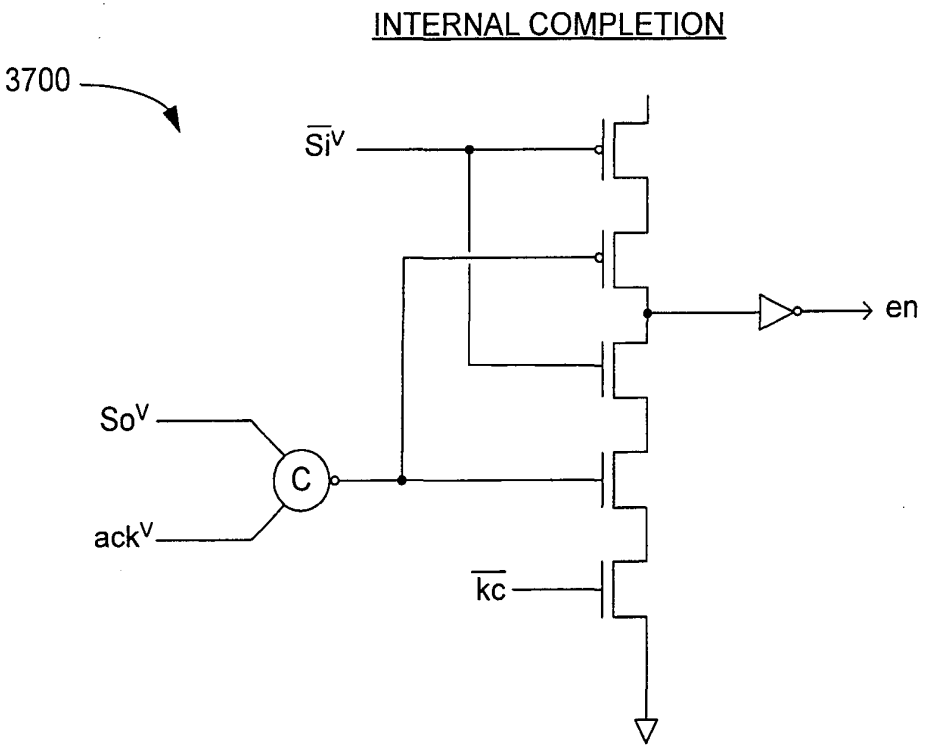

INTERNAL COMPLETION
3700
$\overline{Si}^V$
en
So^V
C
ack^V
$\overline{kc}$

FIG. 37

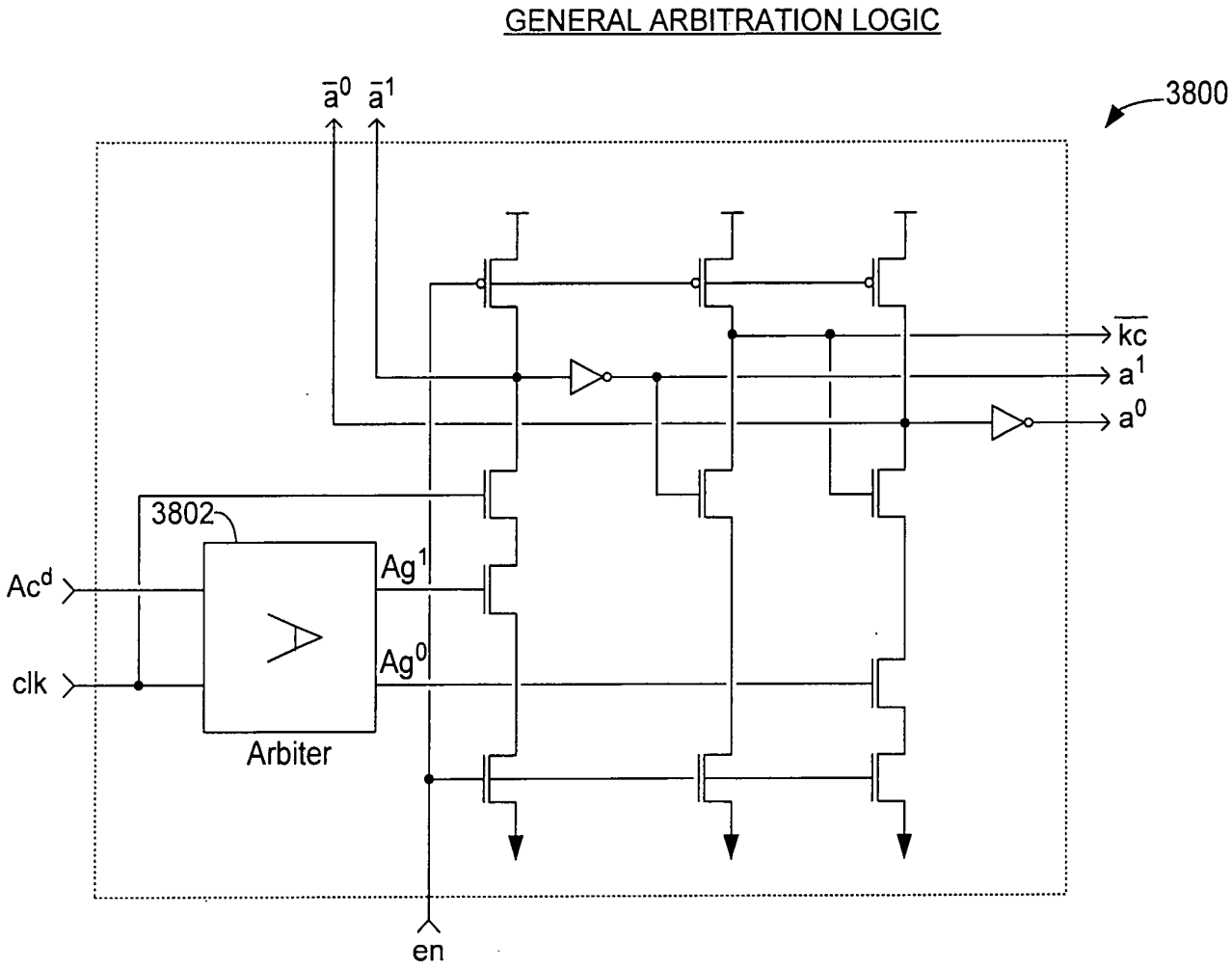

GENERAL ARBITRATION LOGIC
3800
$\bar{a}^0$ $\bar{a}^1$
$\overline{kc}$
a^1
a^0
3802
Ac^d
clk
Ag^1
Ag^0
Arbiter
en

FIG. 38

ARBITRATION LOGIC (clk-vs-Ac^d)

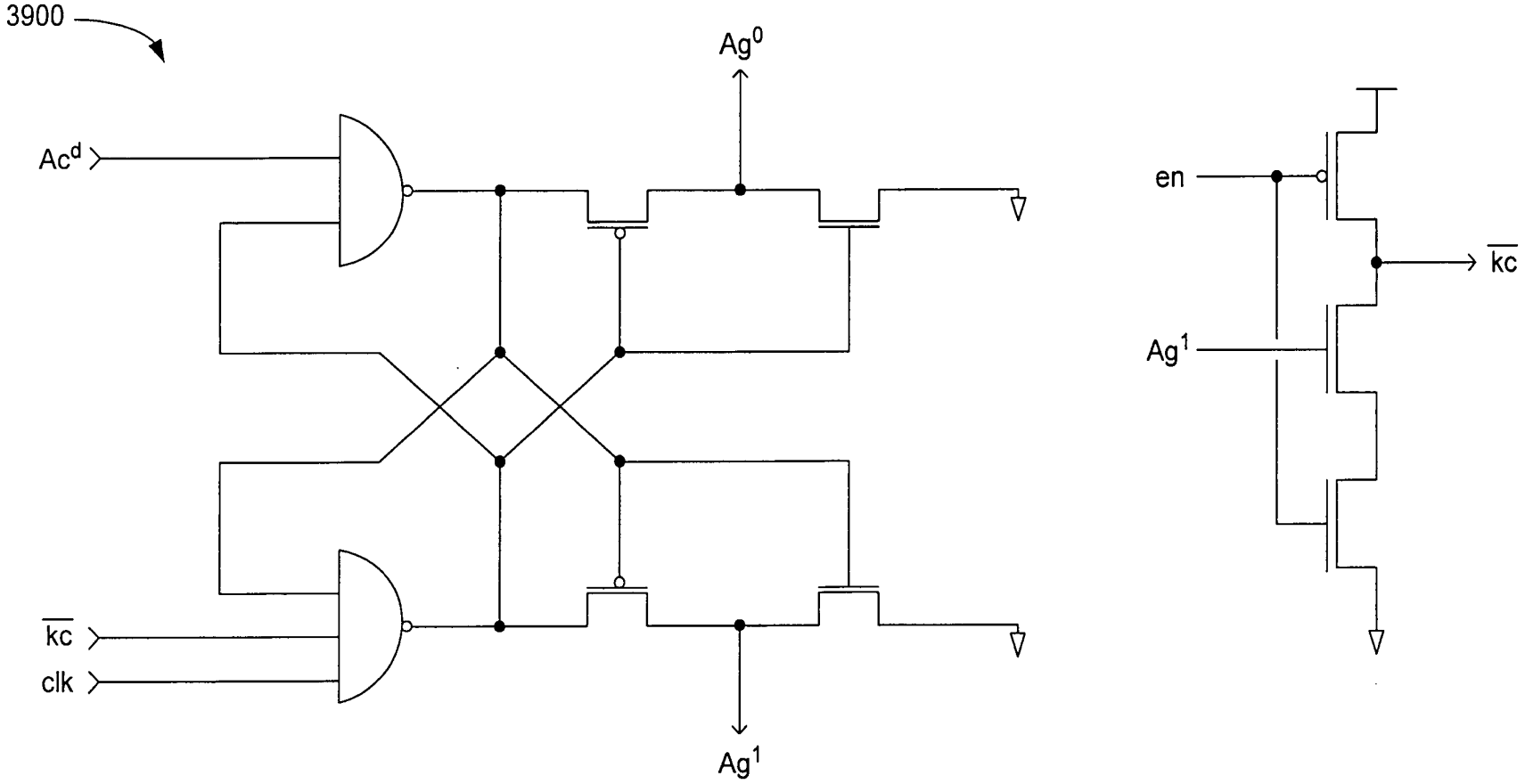


FIG. 39

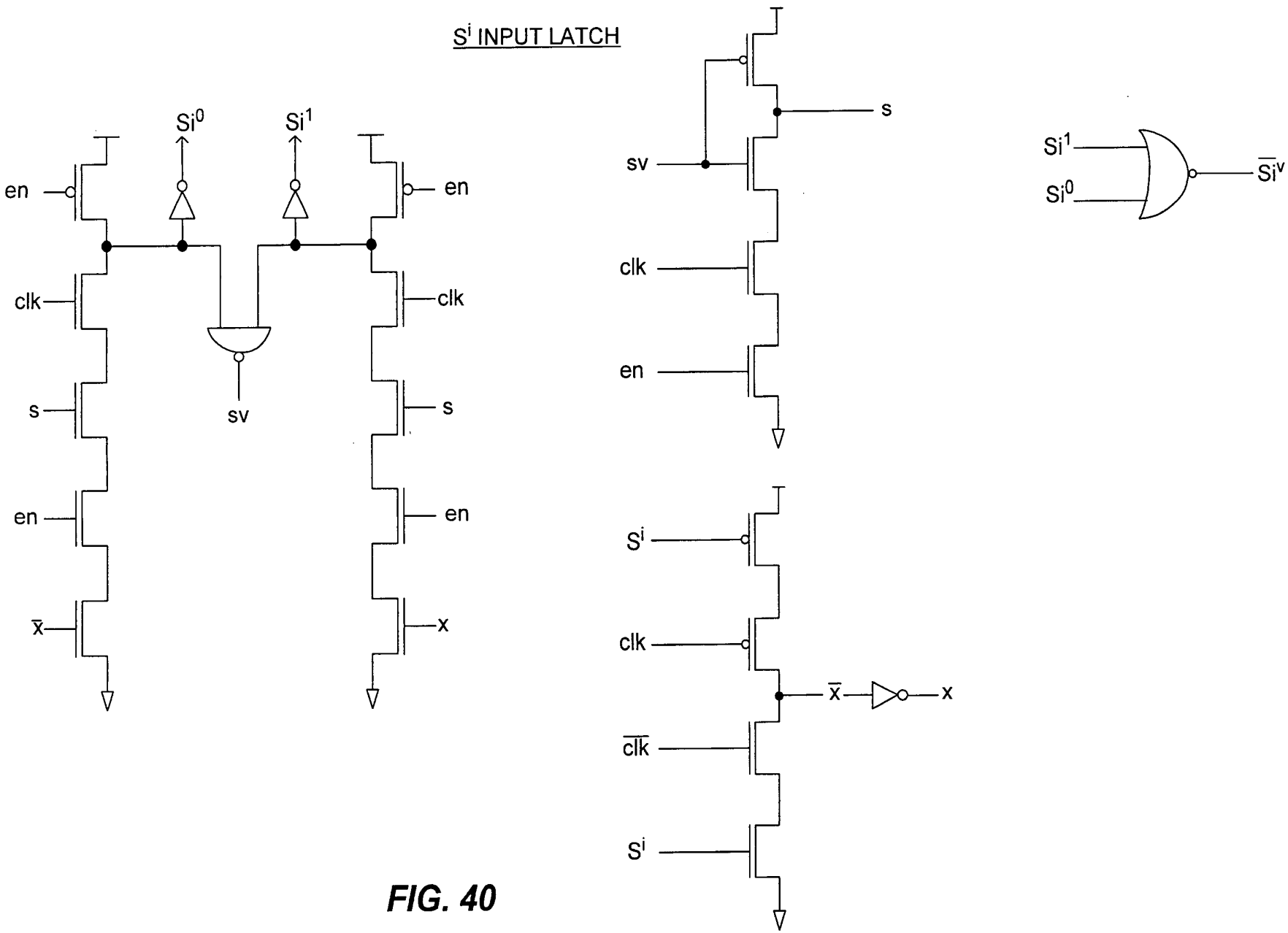

Si INPUT LATCH
Si0
Si1
en
clk
s
sv
en
x
clk
s
en
sv
clk
en
Si
clk
x
clk
Si
Si1
Si0
Siv

FIG. 40

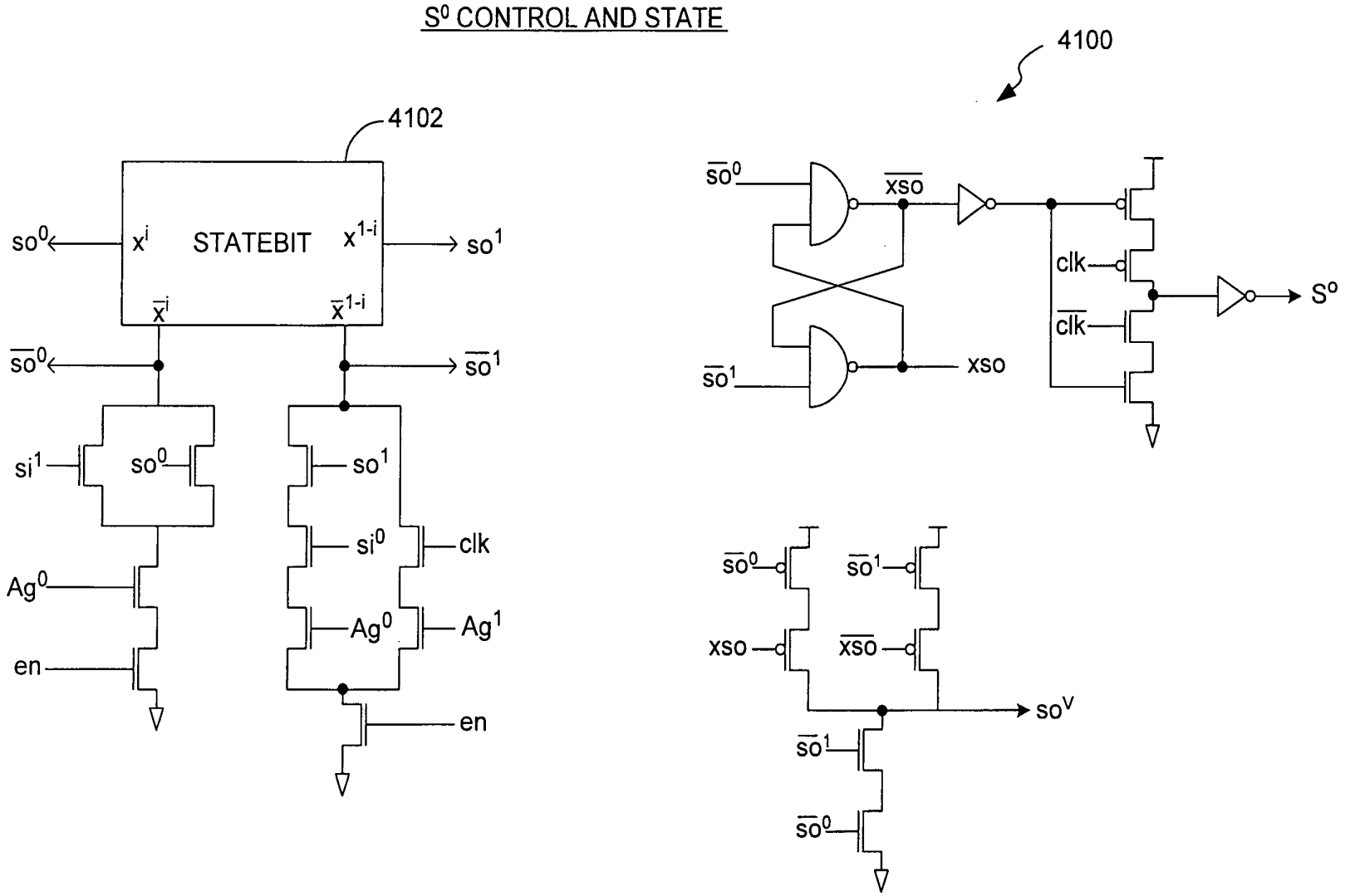

S⁰ CONTROL AND STATE
4100
4102
STATEBIT
so⁰
x^i
x^1-i
so¹
x̄^i
x̄^1-i
so⁰
so¹
si¹
so⁰
so¹
si⁰
clk
Ag⁰
Ag⁰
Ag¹
en
en
so⁰
xso
so¹
xso
clk
clk
S⁰
so⁰
so¹
xso
xso
so^V
so¹
so⁰

FIG. 41

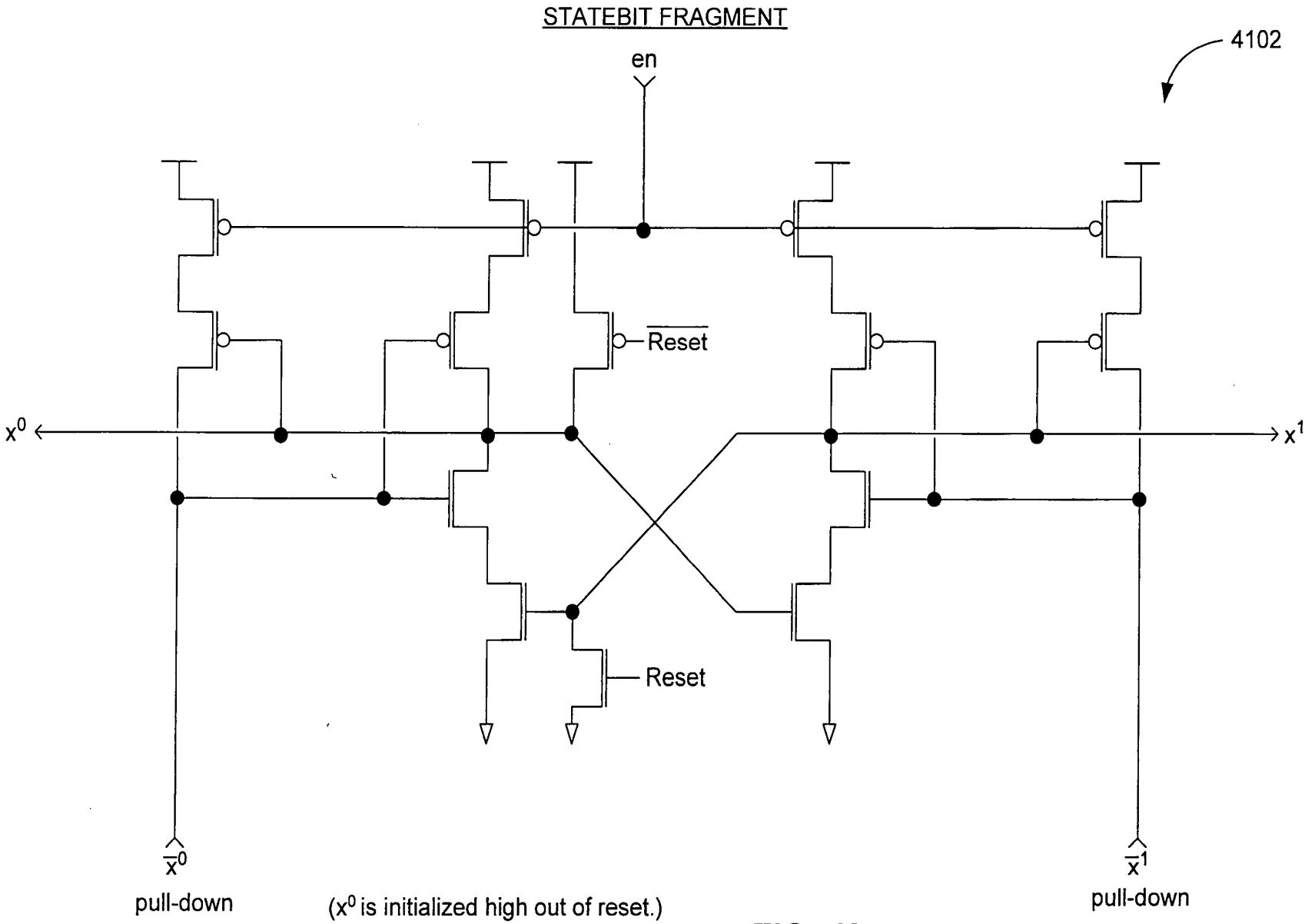

STATEBIT FRAGMENT
4102
en
Reset
Reset
x^0
x^1
$\overline{x}^0$
pull-down
(x^0 is initialized high out of reset.)
$\overline{x}^1$
pull-down

FIG. 42

Ac ACKNOWLEDGE CONTROL

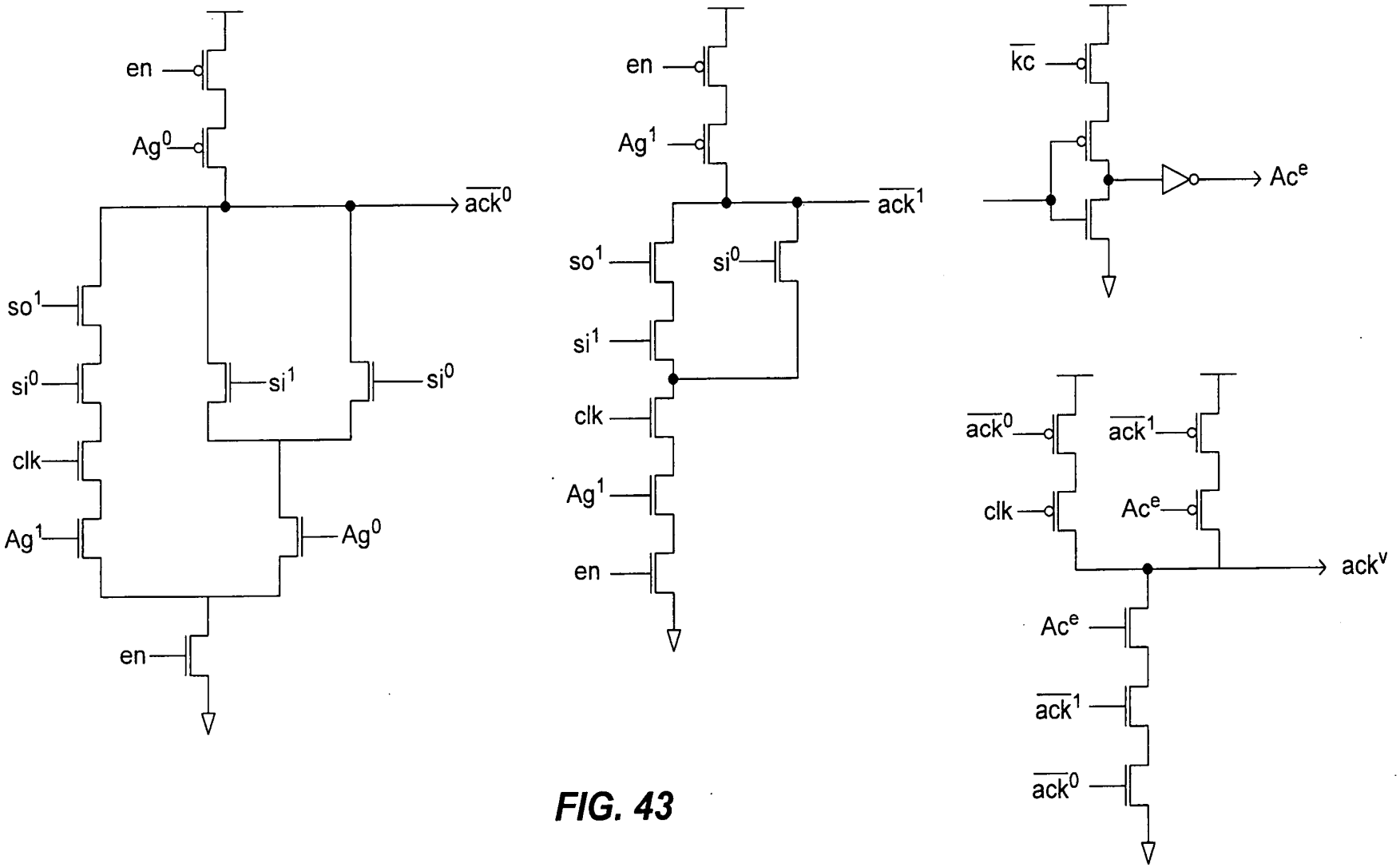


FIG. 43

A2S DATAPATH TRANSFER UNIT

(1-OF-2 VERSION)

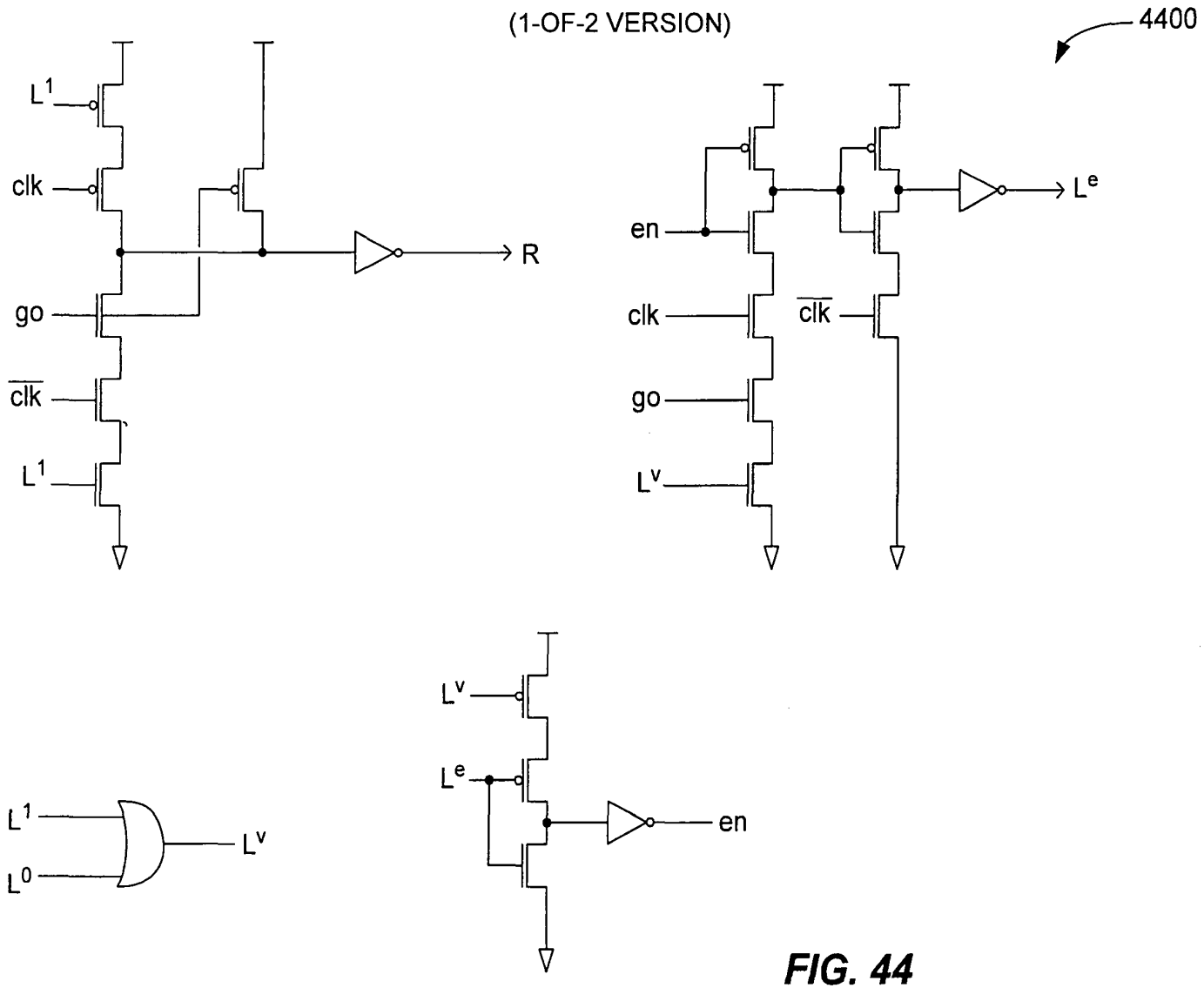


FIG. 44

S2A DATAPATH TRANSFER UNIT

4500

DTU-S2A-BIT ☰
* [[¬clk]; [clk];
[go → R! L
[] else → skip
]]

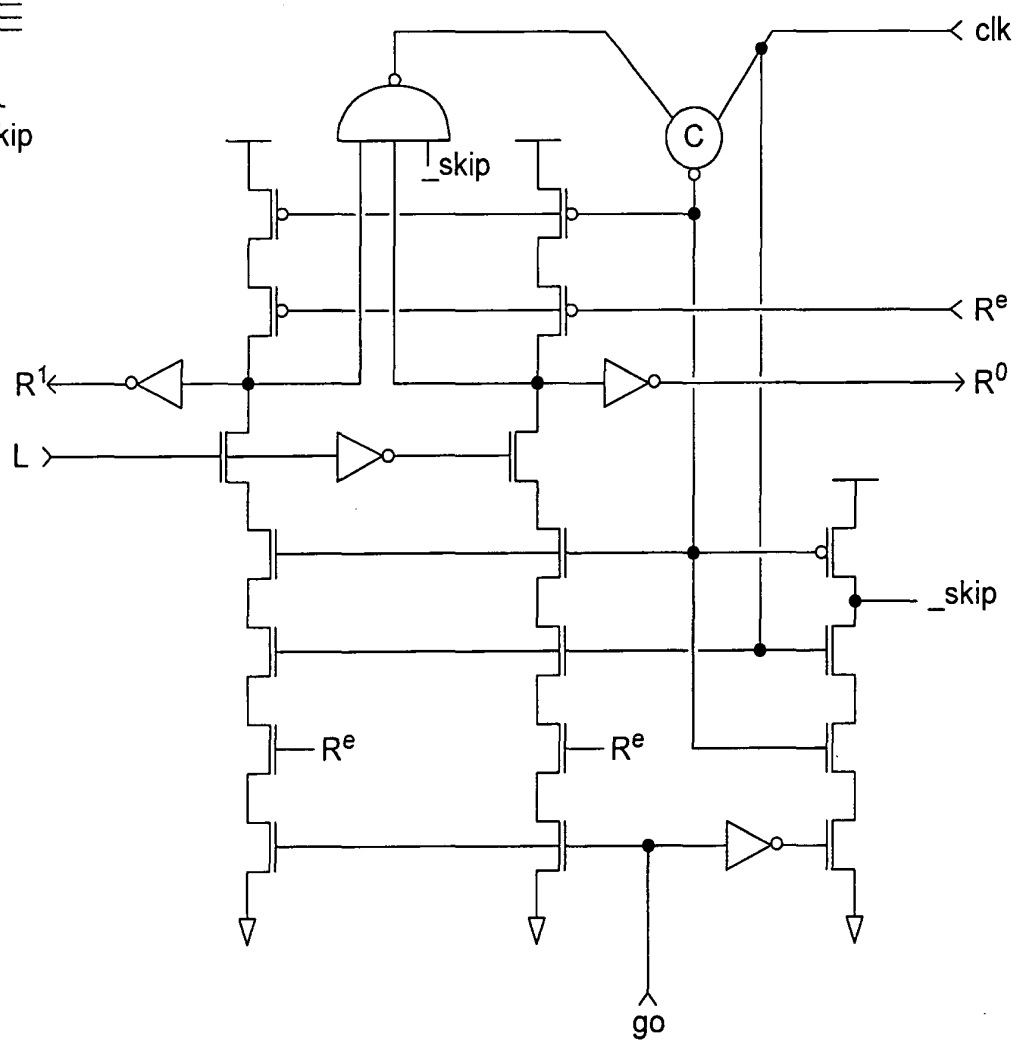


FIG. 45

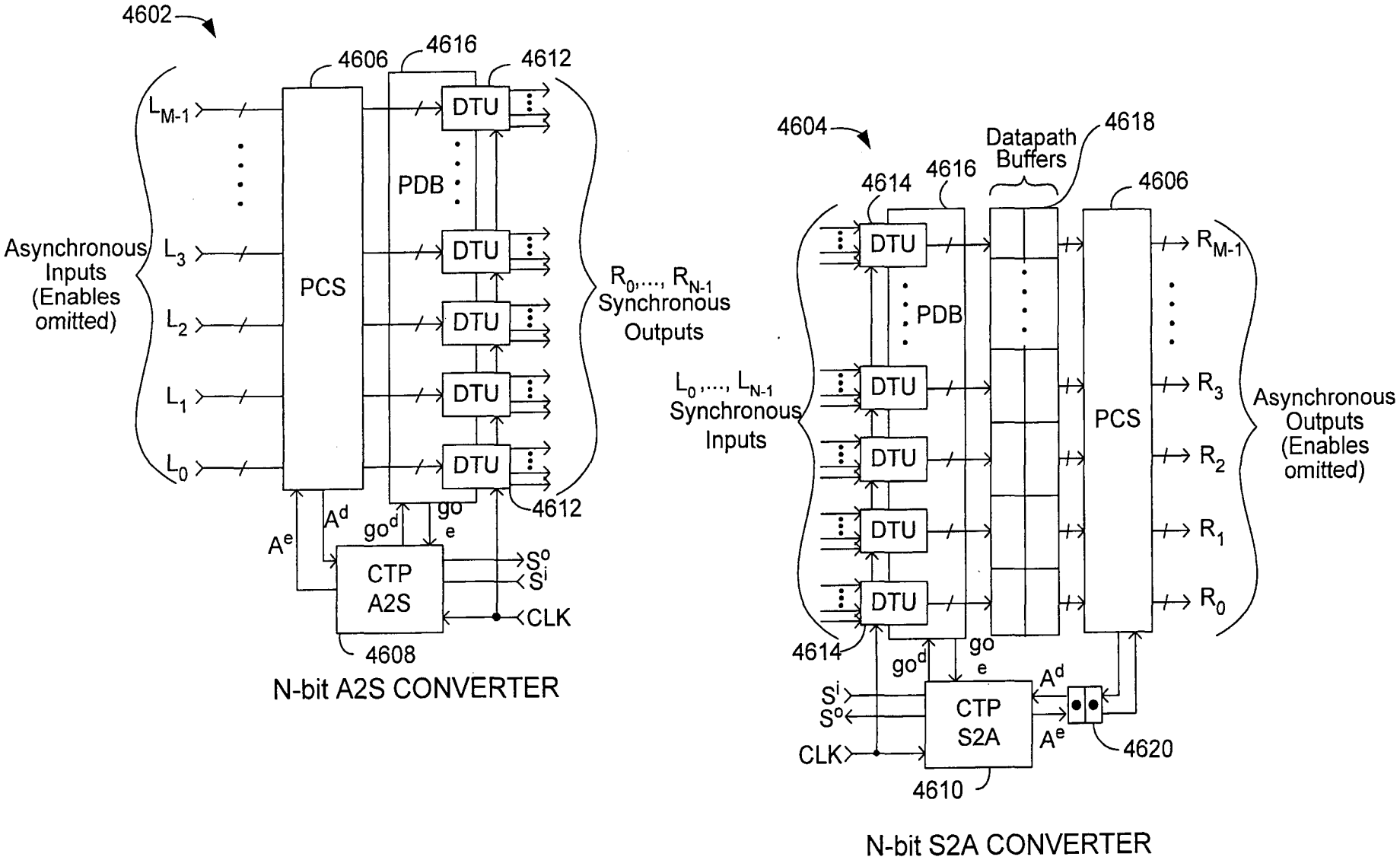

4602
4606
4616
4612
DTU
PDB
PCS
Asynchronous Inputs (Enables omitted)
L_{M-1}
L_3
L_2
L_1
L_0
$R_0,\ldots, R_{N-1}$ Synchronous Outputs
4612
A^d
go^d
go^e
A^e
CTP A2S
S^o
S^i
CLK
4608
N-bit A2S CONVERTER
4604
4614
4616
Datapath Buffers
4618
4606
$L_0,\ldots, L_{N-1}$ Synchronous Inputs
R_{M-1}
R_3
R_2
R_1
R_0
Asynchronous Outputs (Enables omitted)
4614
go^d
go^e
A^d
A^e
S^i
S^o
CLK
CTP S2A
4620
4610
N-bit S2A CONVERTER

FIG. 46

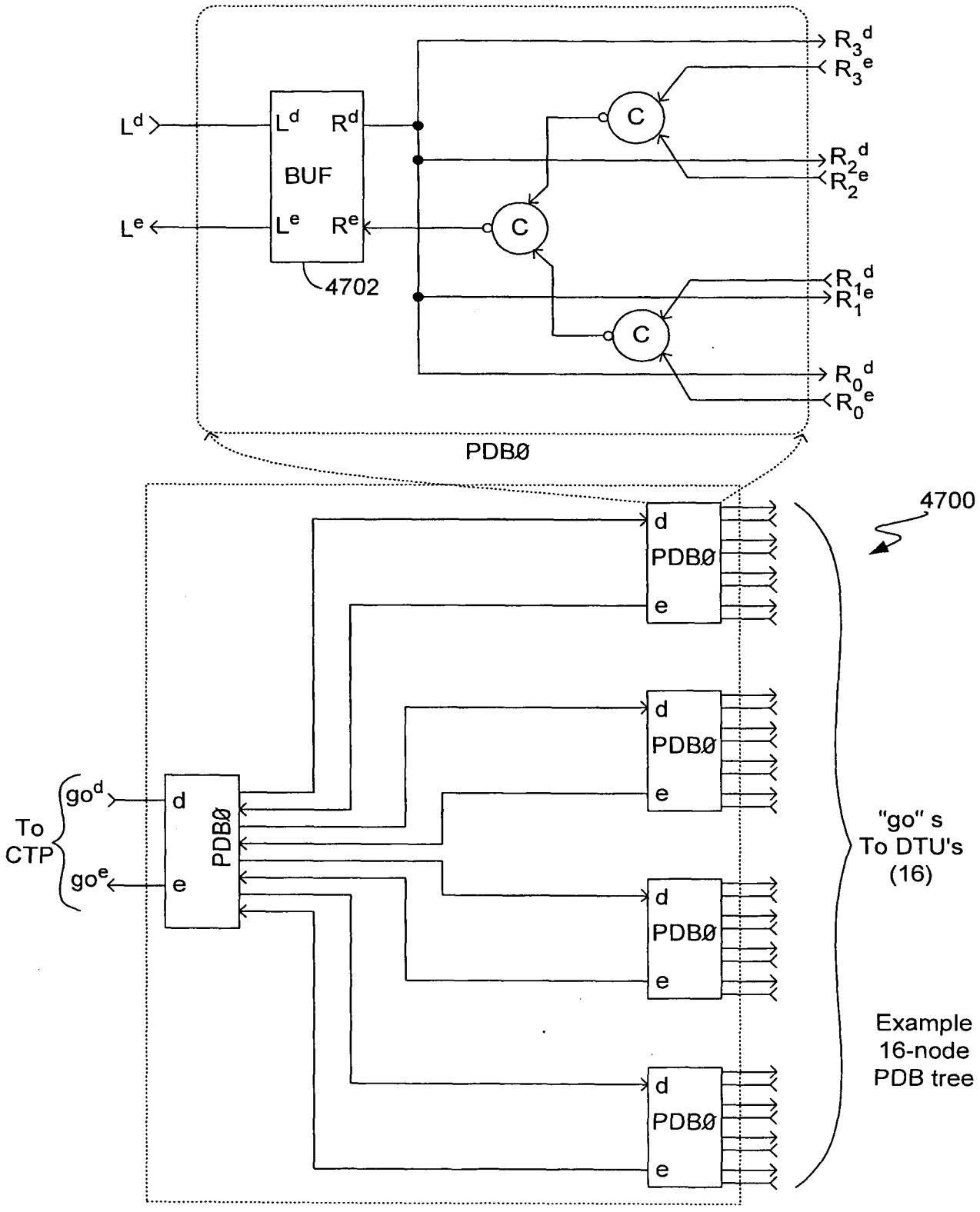

L^d
L^e
L^d R^d
BUF
L^e R^e
4702
C
C
C
R_3^d
R_3^e
R_2^d
R_2^e
R_1^d
R_1^e
R_0^d
R_0^e
PDBØ
4700
d
PDBØ
e
go^d
go^e
To CTP
"go" s To DTU's (16)
Example 16-node PDB tree

FIG. 47

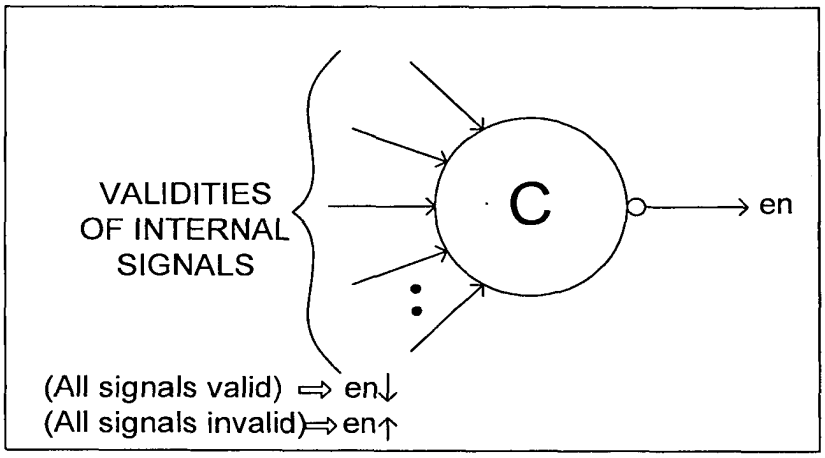

VALIDITIES
OF INTERNAL
SIGNALS
C
en
(All signals valid) ⇨ en↓
(All signals invalid) ⇨ en↑

FIG. 48

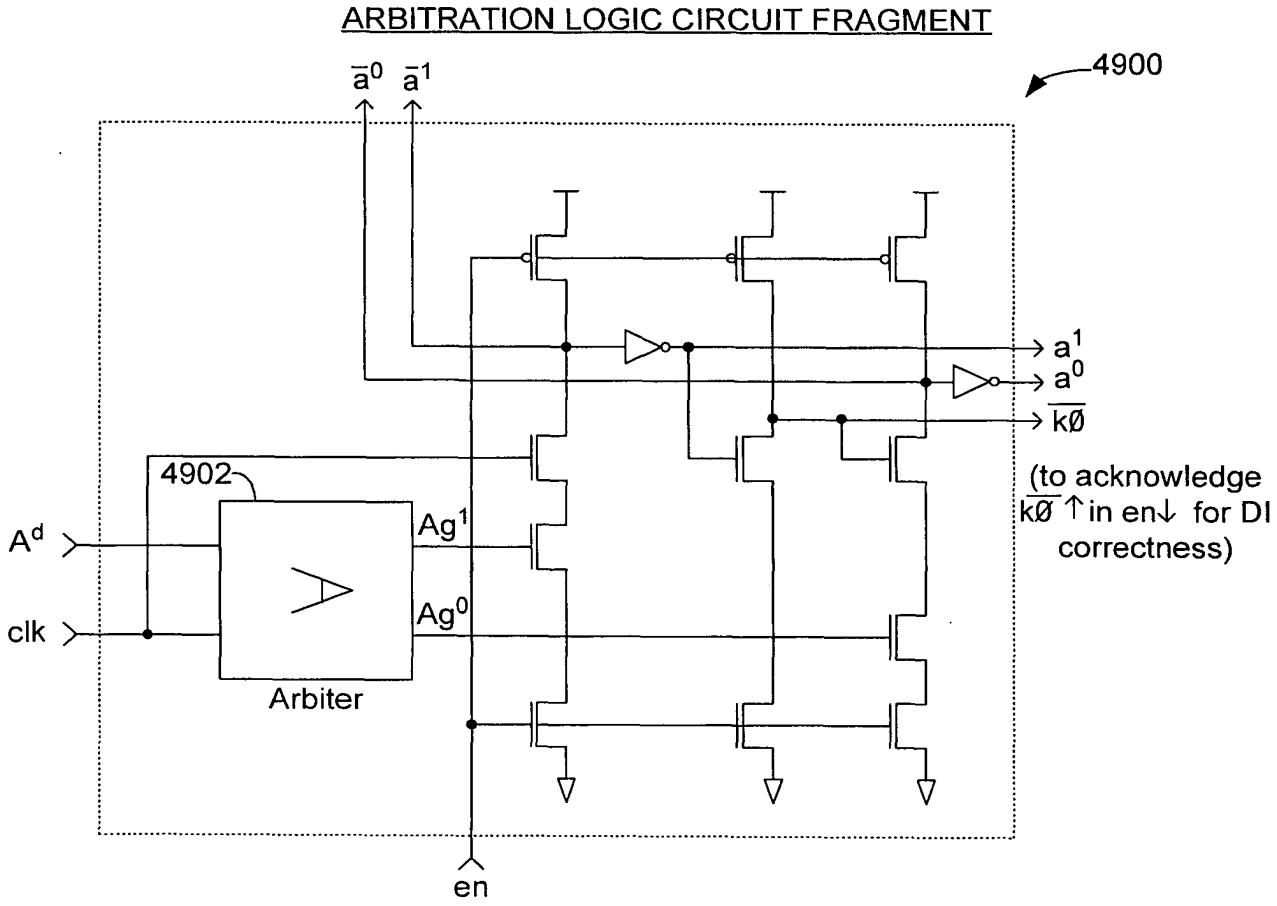

ARBITRATION LOGIC CIRCUIT FRAGMENT
4900
$\bar{a}^0$ $\bar{a}^1$
a^1
a^0
$\overline{k\emptyset}$
(to acknowledge $\overline{k\emptyset}$ ↑ in en↓ for DI correctness)
4902
A^d
clk
Ag^1
Ag^0
Arbiter
en

FIG. 49

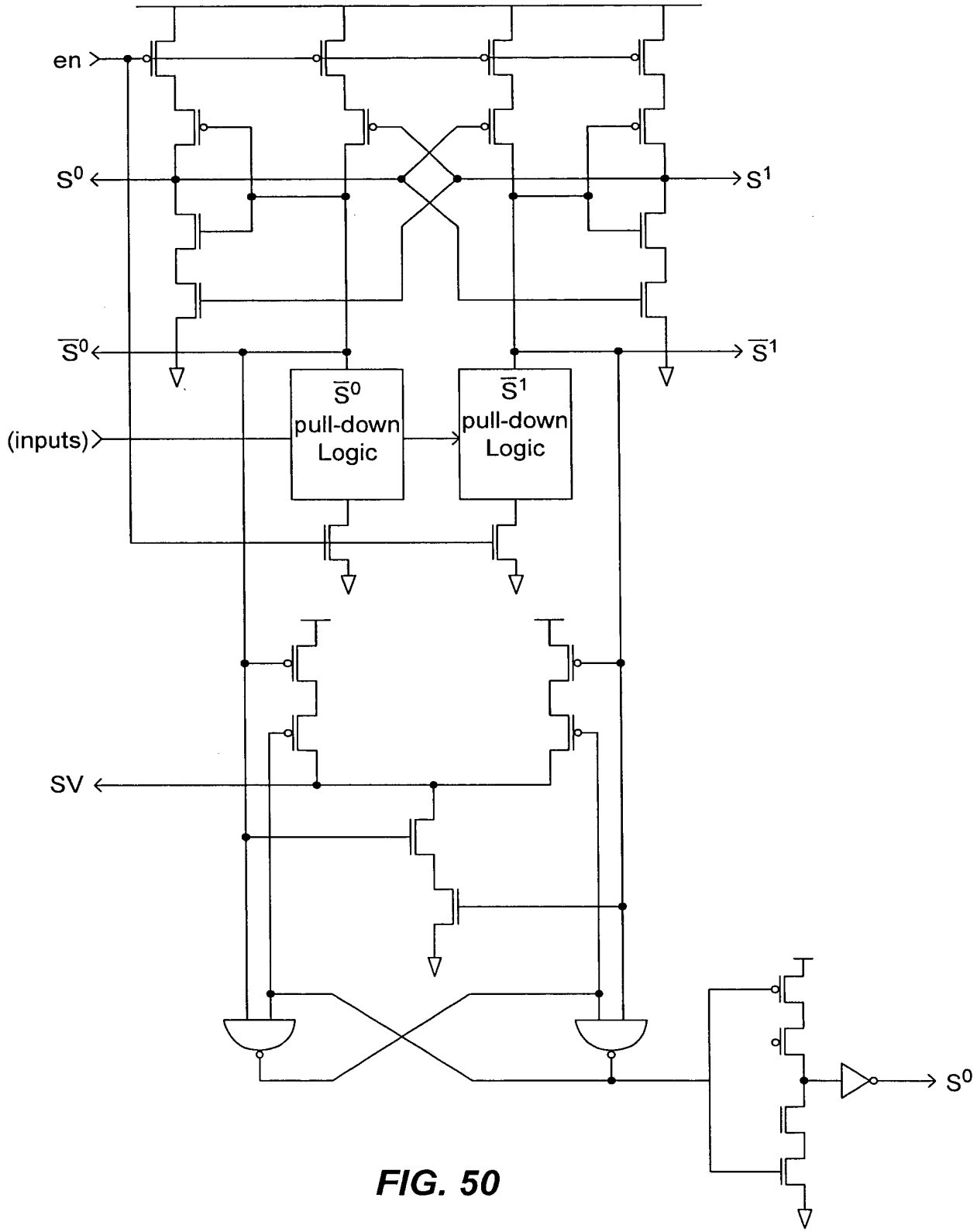

en
S^0
S^1
$\overline{S}^0$
$\overline{S}^1$
(inputs)
$\overline{S}^0$ pull-down Logic
$\overline{S}^1$ pull-down Logic
SV
S^0

FIG. 50

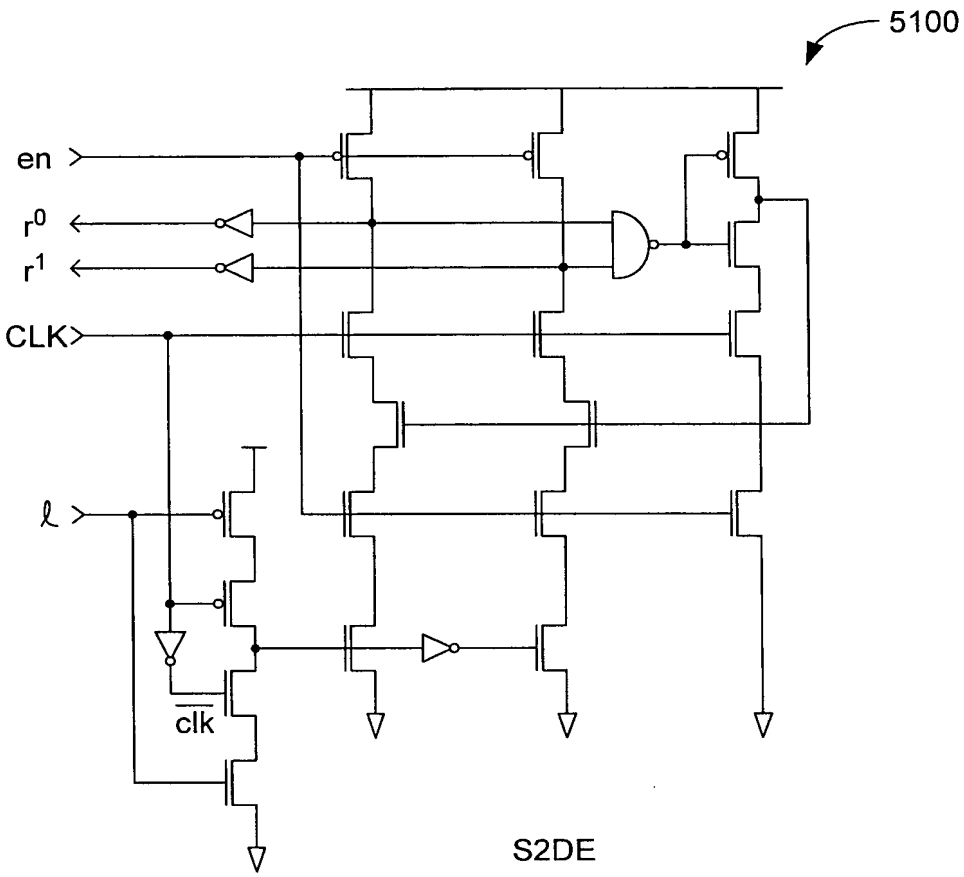


FIG. 51

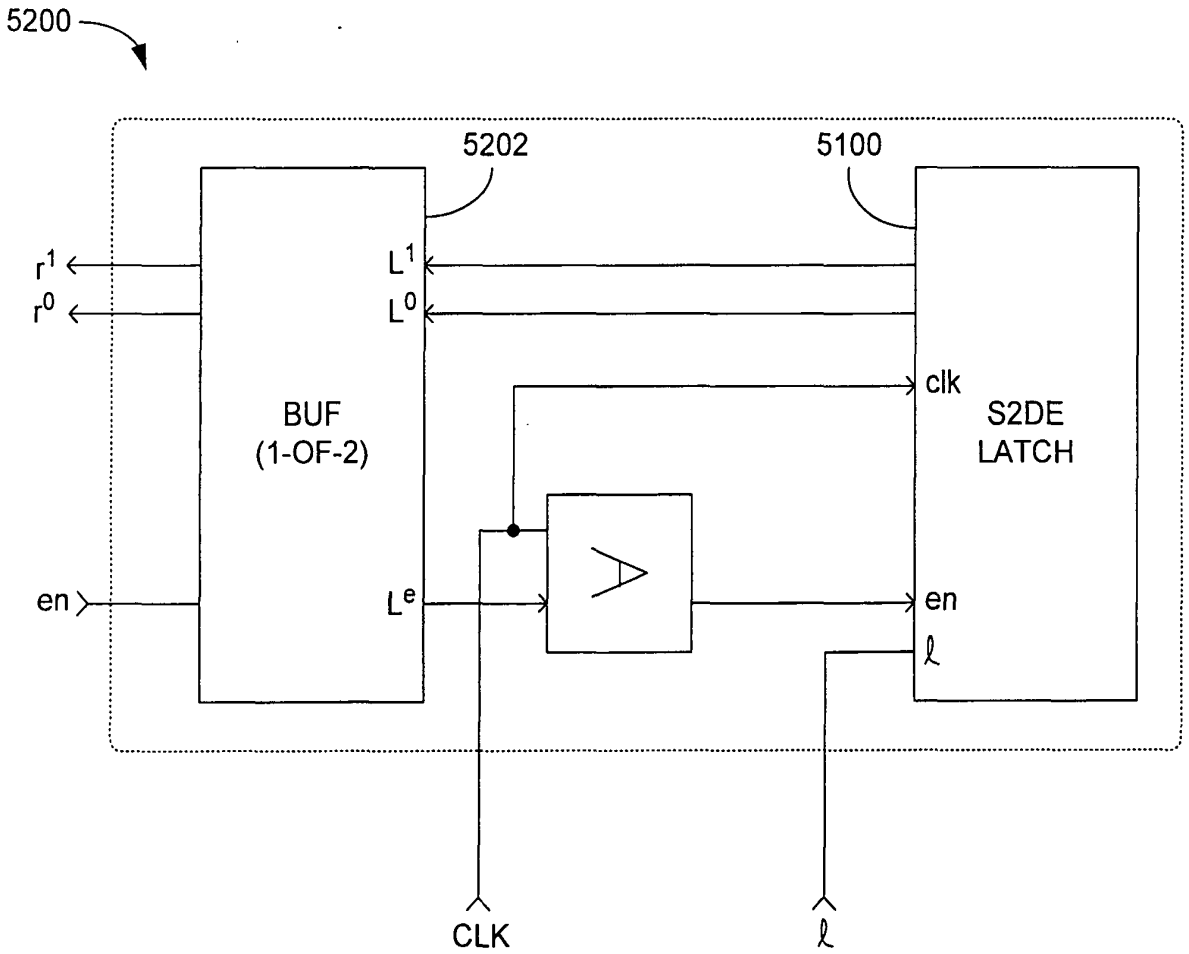

5200
5202
5100
r^1
r^0
L^1
L^0
clk
BUF
(1-OF-2)
S2DE
LATCH
en
L^e
en
ℓ
CLK
ℓ

FIG. 52

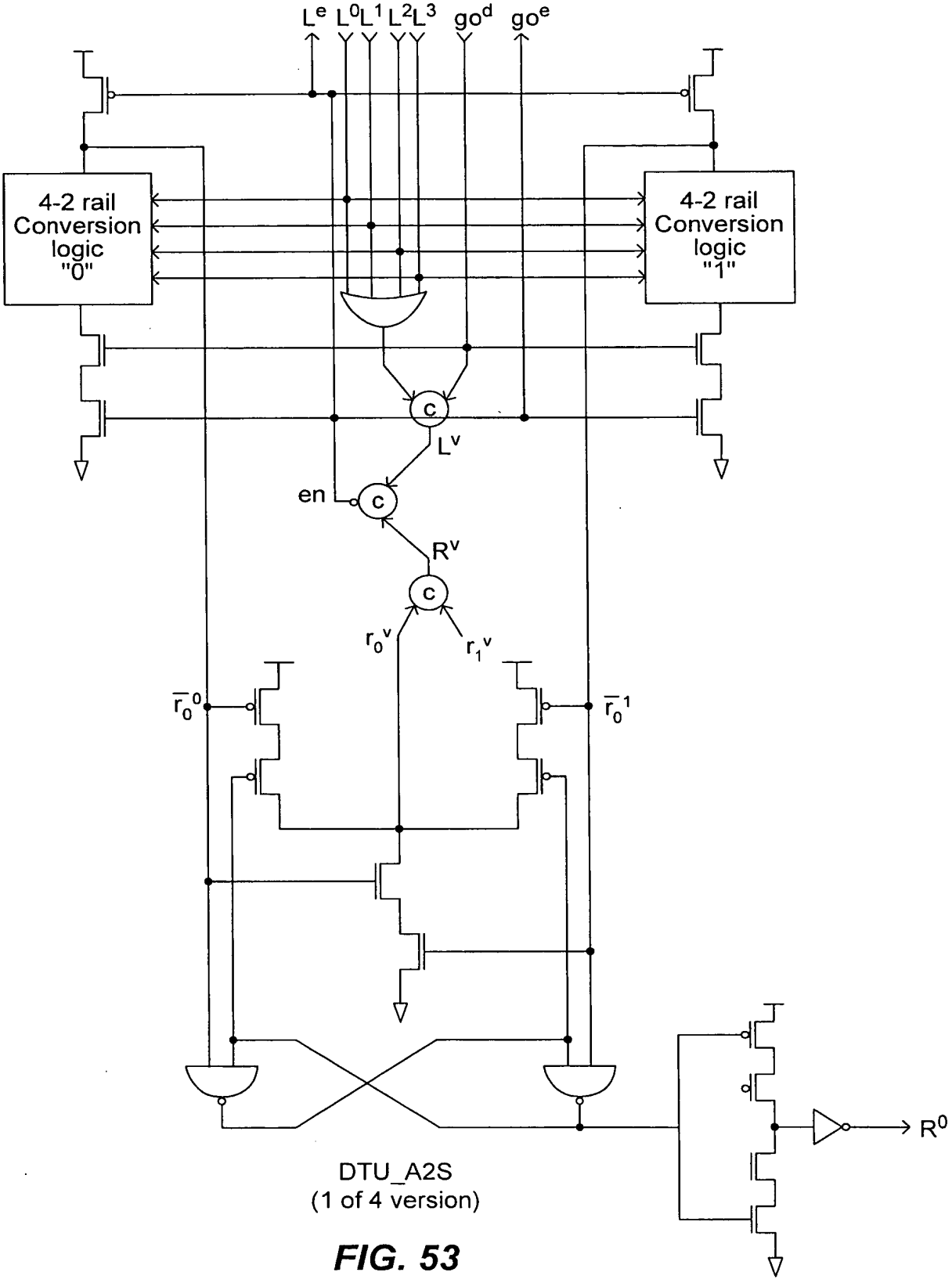
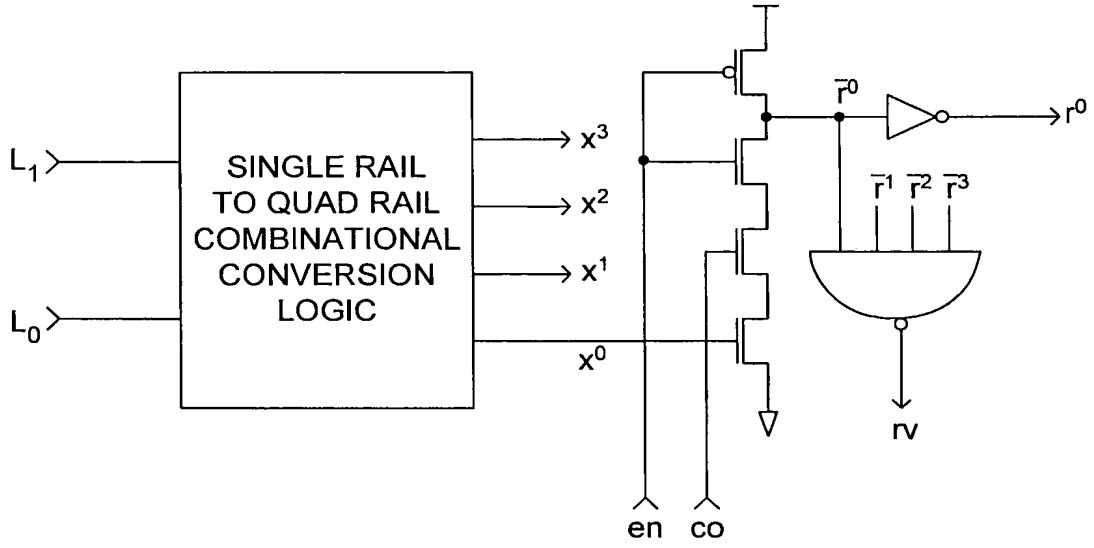

L^e L^0L^1 L^2L^3 go^d go^e
4-2 rail Conversion logic "0"
4-2 rail Conversion logic "1"
C
L^v
en
C
R^v
C
r_0^v
r_1^v
$\overline{r_0}^0$
$\overline{r_0}^1$
R^0
DTU_A2S
(1 of 4 version)

FIG. 53



S2Q SAMPLER

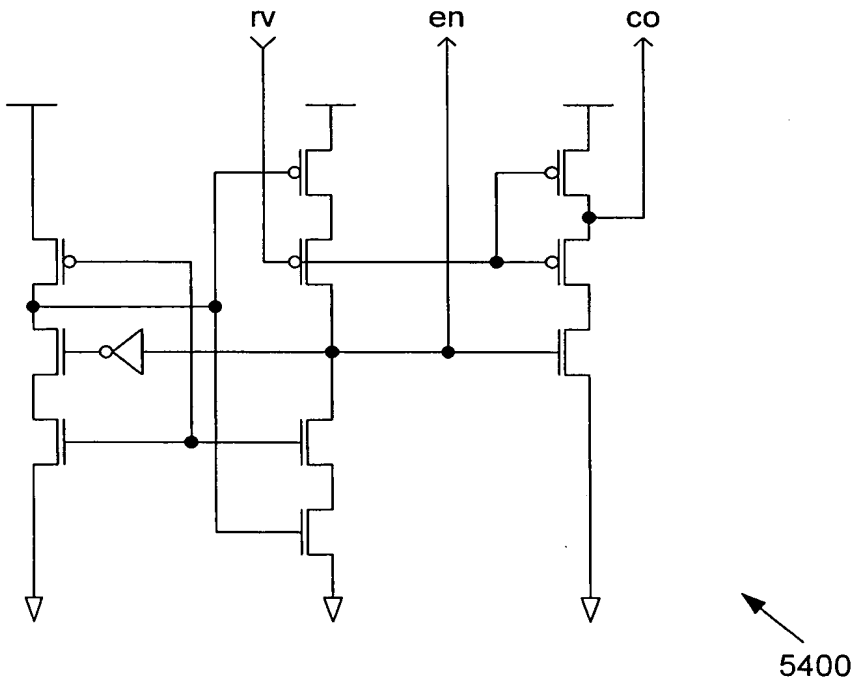
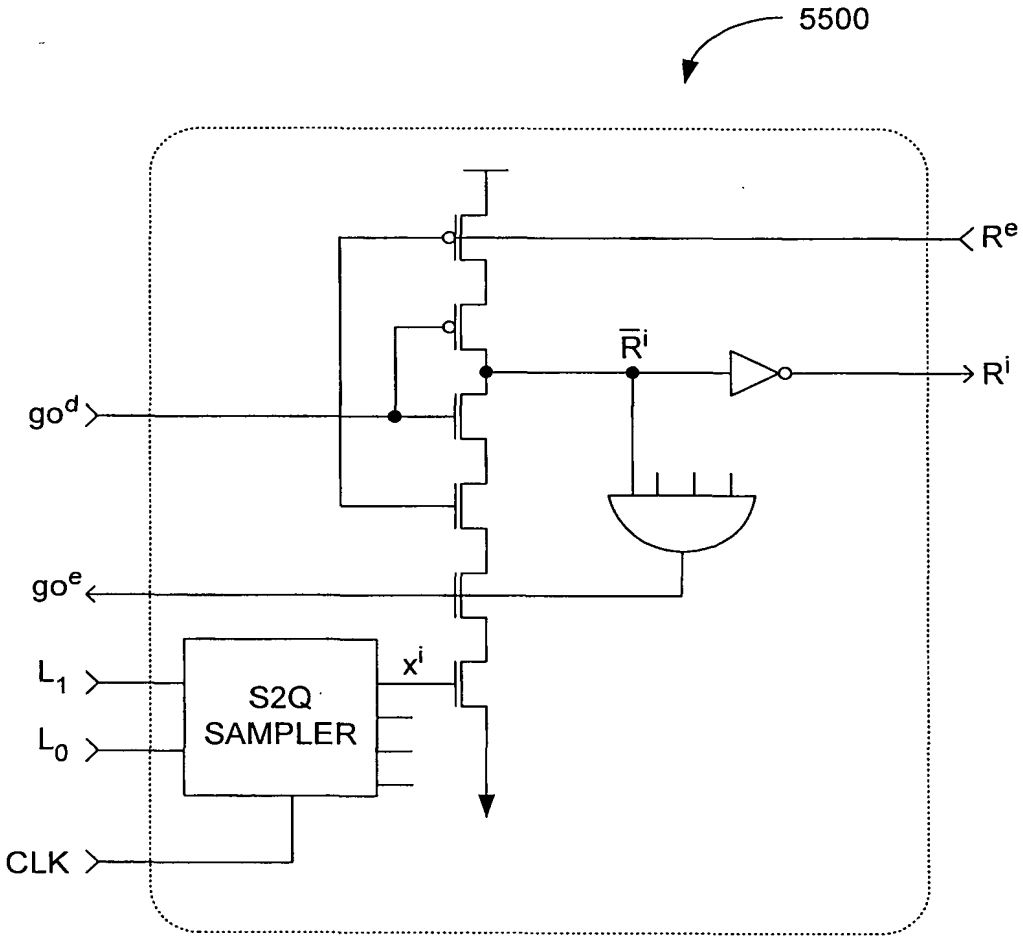


FIG. 54



DTU-S2A
(1-of-4 version)

FIG. 55

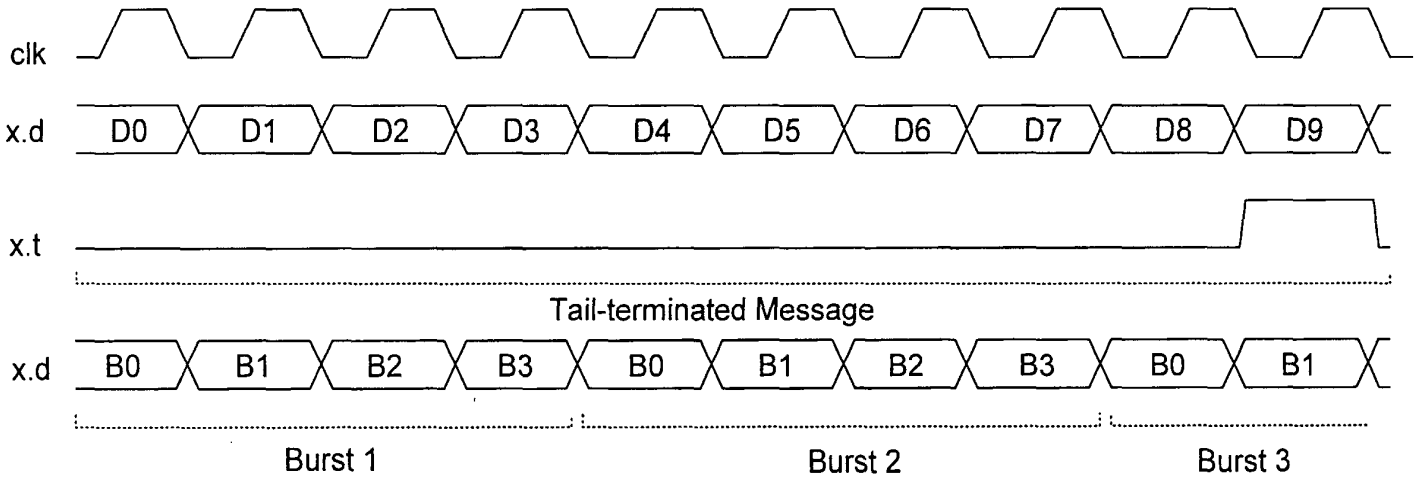
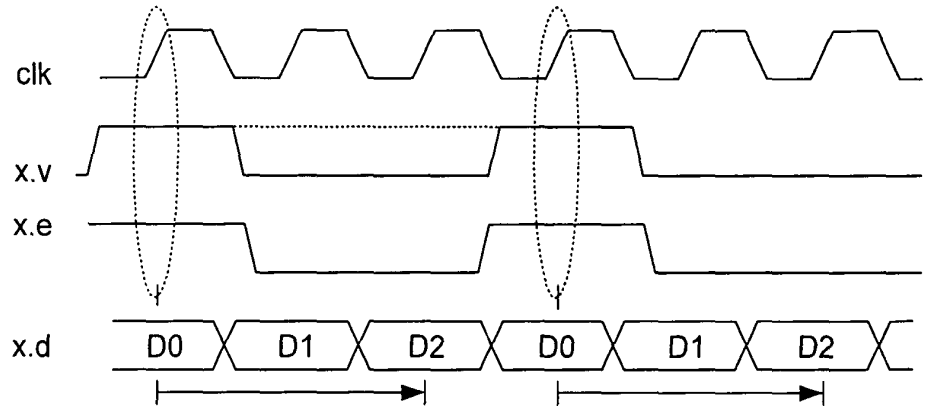
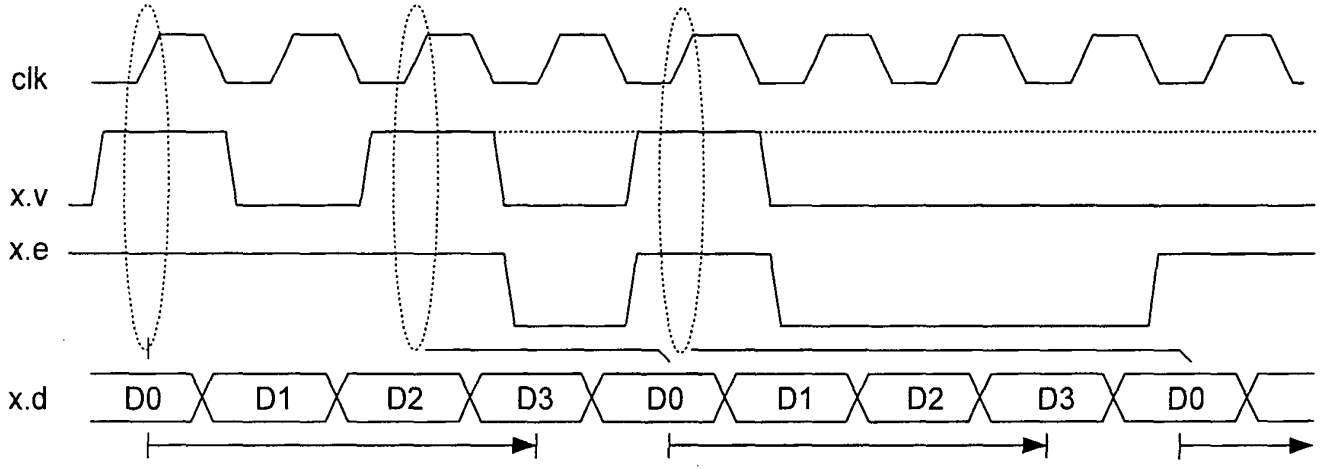


FIG. 56 Message segmentation MAX_LEN=4



Non-pipelined 3-word burst transfers



Pipelined 4-word burst transfers
(Receiver grants at most one outstanding transfer)

FIG. 57 Burst Transfer Protocol

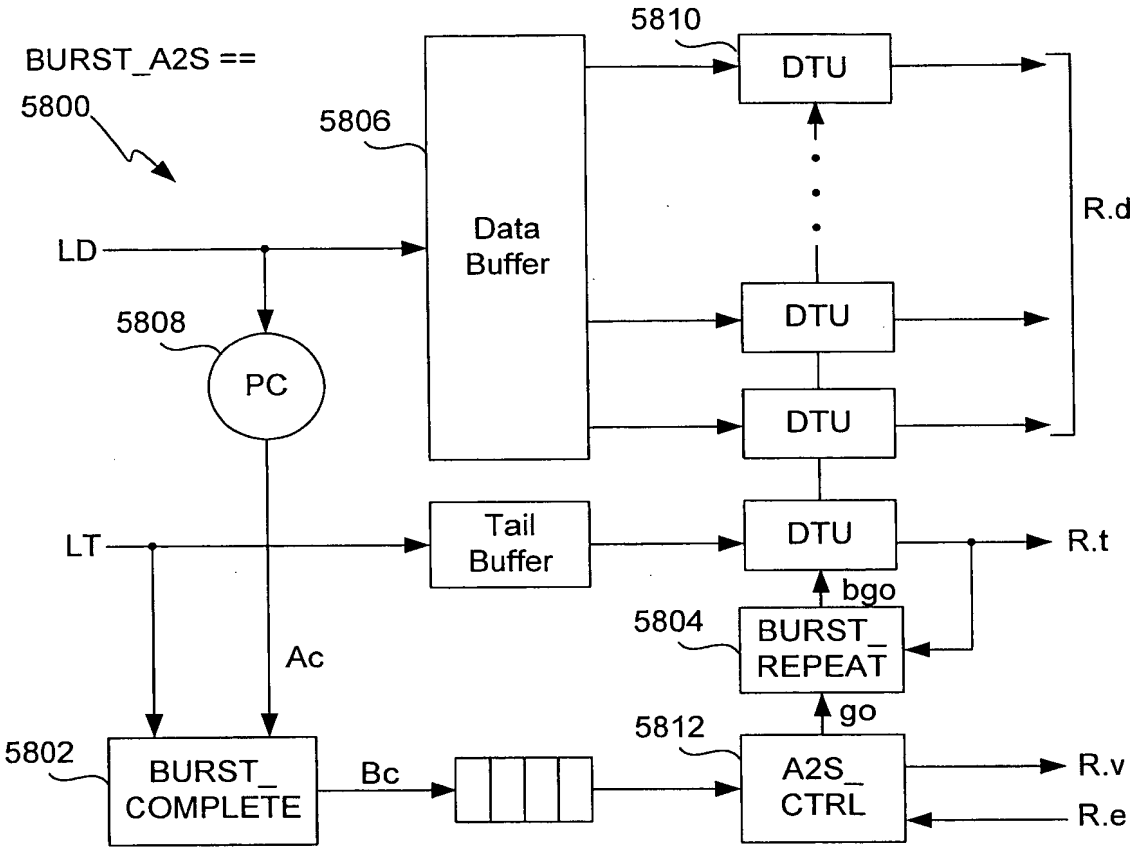

BURST_A2S ==
5800
5810
DTU
5806
Data Buffer
R.d
LD
5808
PC
DTU
DTU
LT
Tail Buffer
DTU
R.t
bgo
5804
BURST_ REPEAT
Ac
go
5812
5802
BURST_ COMPLETE
Bc
A2S_ CTRL
R.v
R.e

FIG. 58

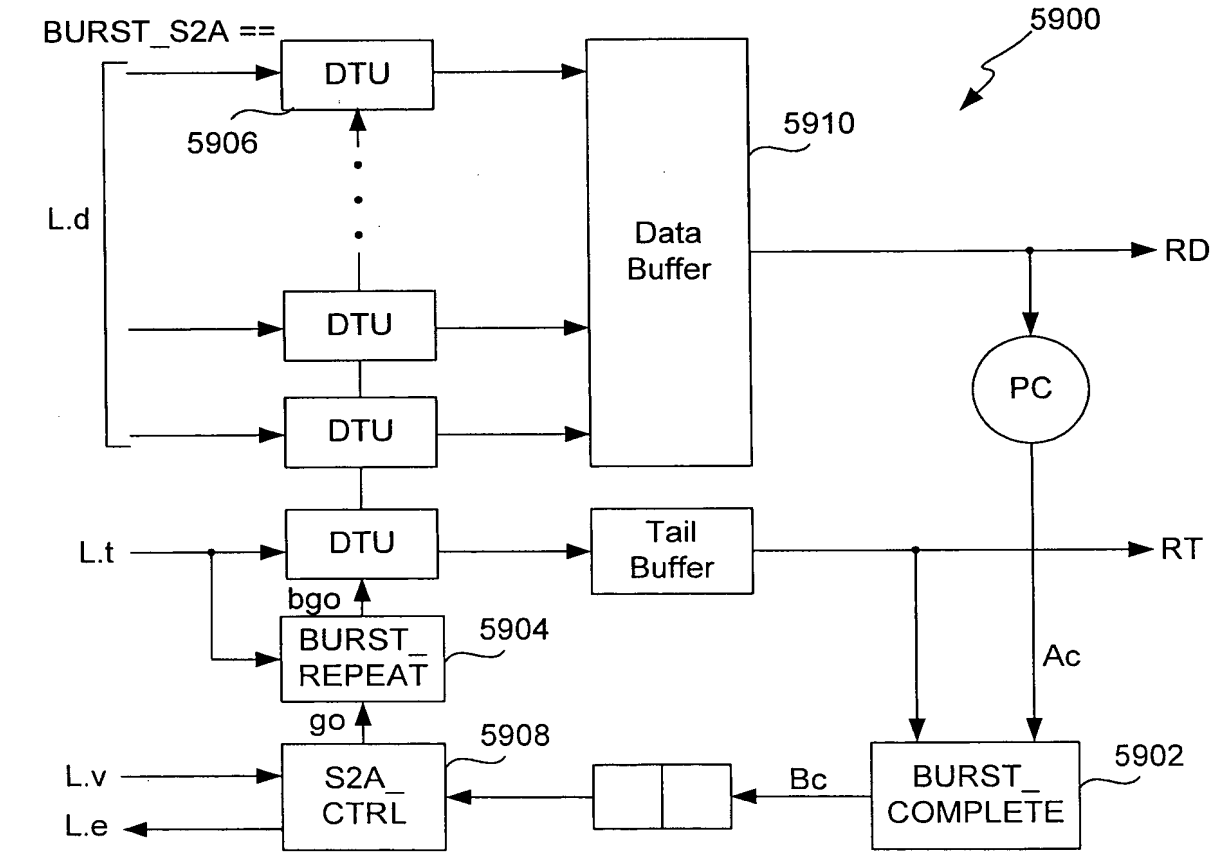

BURST_S2A ==
5900
DTU
5906
5910
L.d
Data Buffer
RD
DTU
PC
DTU
L.t
DTU
Tail Buffer
RT
bgo
BURST_ REPEAT
5904
Ac
go
5908
5902
L.v
S2A_ CTRL
Bc
BURST_ COMPLETE
L.e

FIG. 59

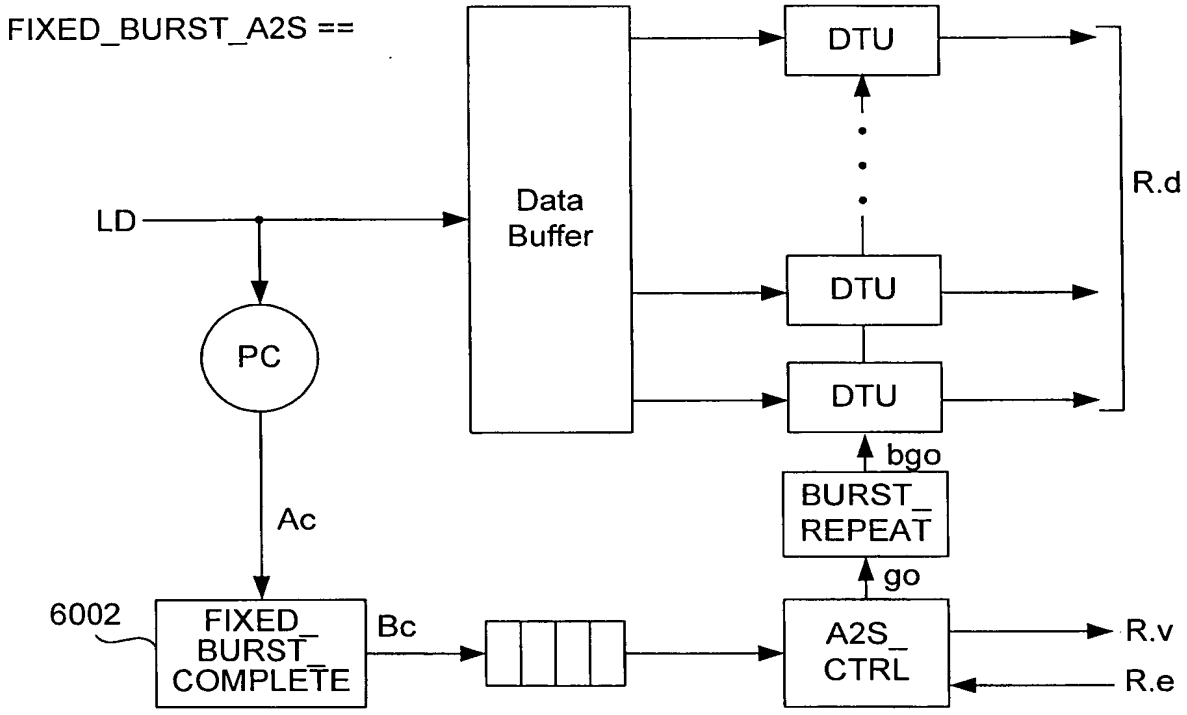

FIXED_BURST_A2S ==
DTU
Data Buffer
R.d
LD
PC
DTU
DTU
bgo
BURST_ REPEAT
Ac
go
6002
FIXED_ BURST_ COMPLETE
Bc
A2S_ CTRL
R.v
R.e

FIG. 60

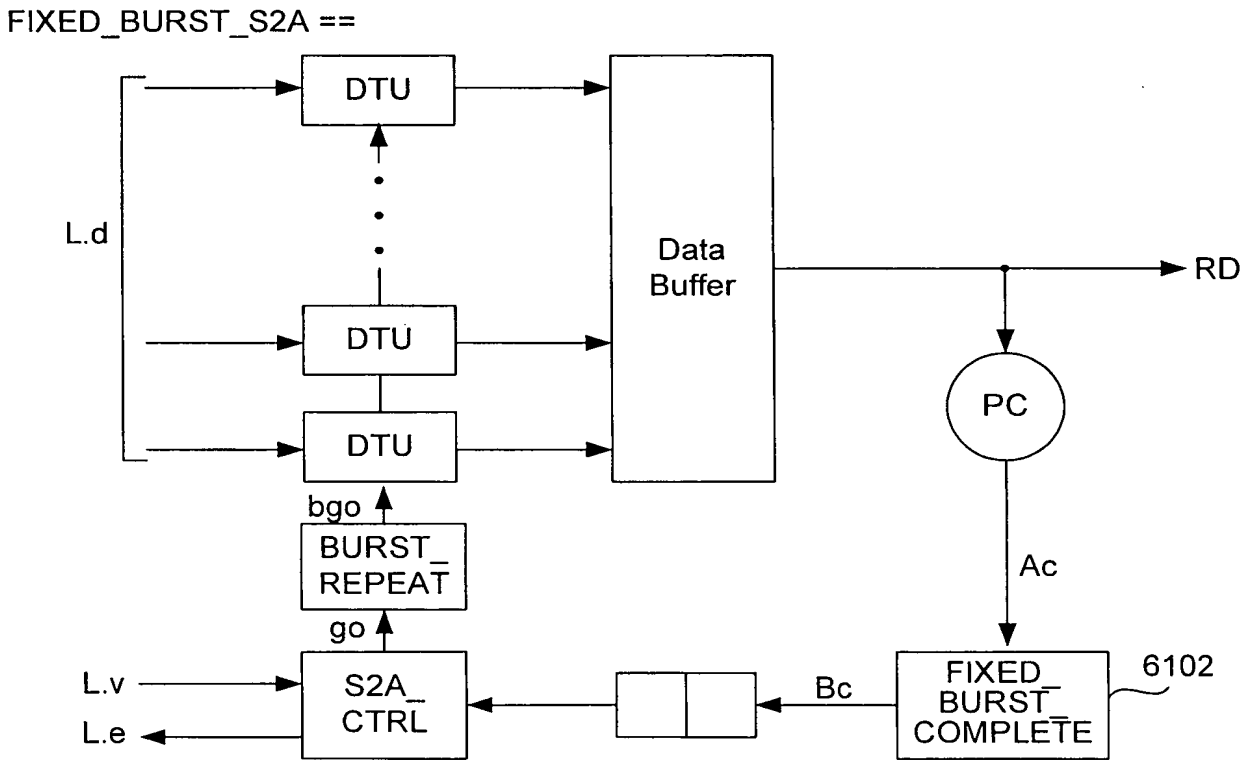
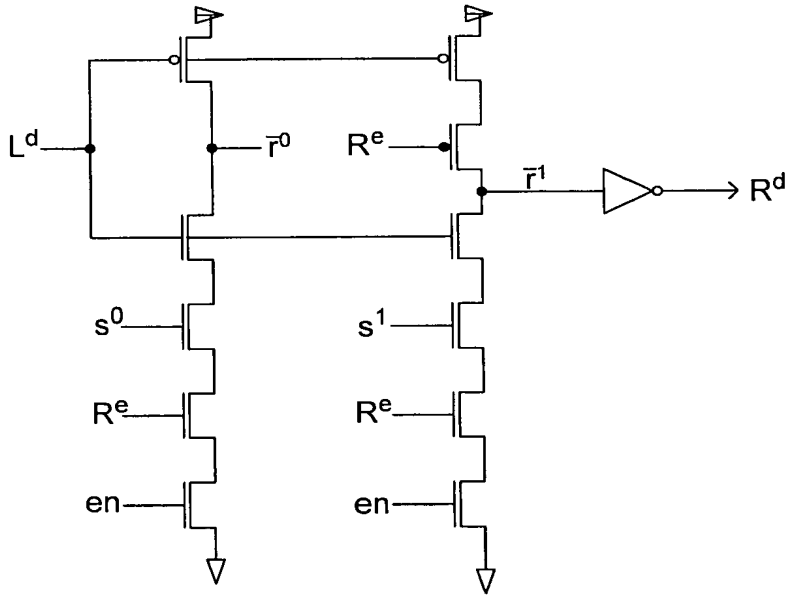
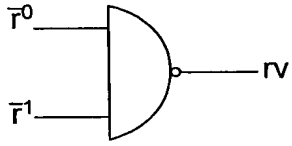

FIXED_BURST_S2A ==
DTU
L.d
Data Buffer
RD
DTU
PC
DTU
bgo
BURST_ REPEAT
Ac
go
L.v
L.e
S2A_ CTRL
Bc
FIXED_ BURST_ COMPLETE
6102

FIG. 61

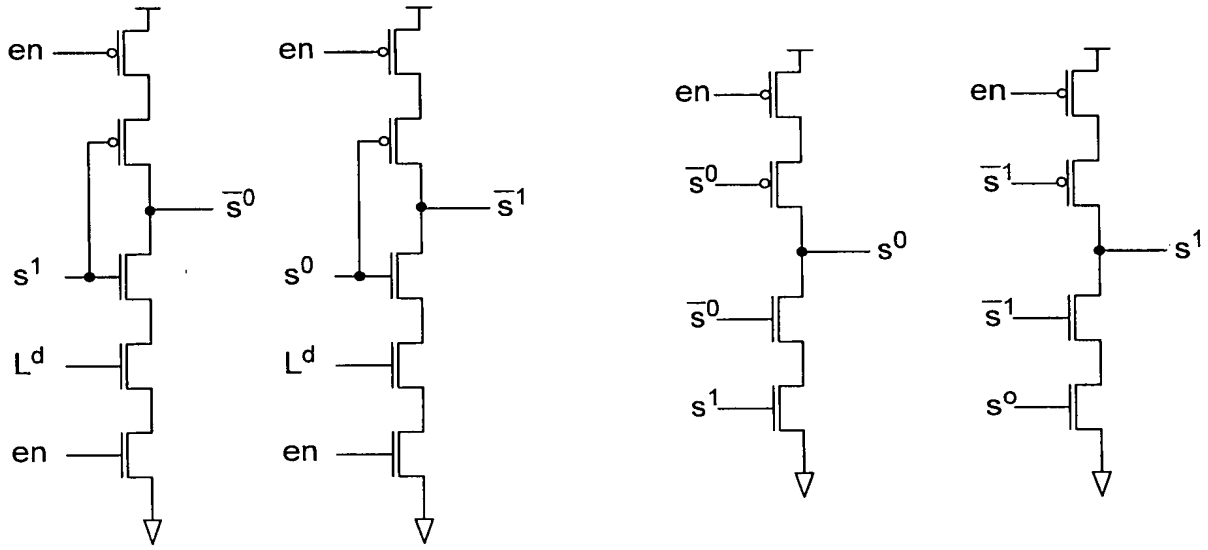
DECIMATE2_1 of 1

Weak-condition output logic:

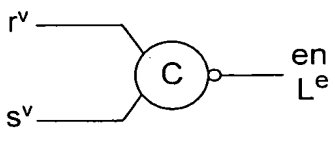




State Logic:



Acknowledge Logic:



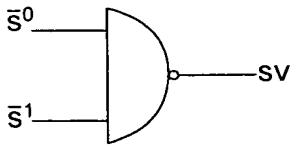


FIG. 62

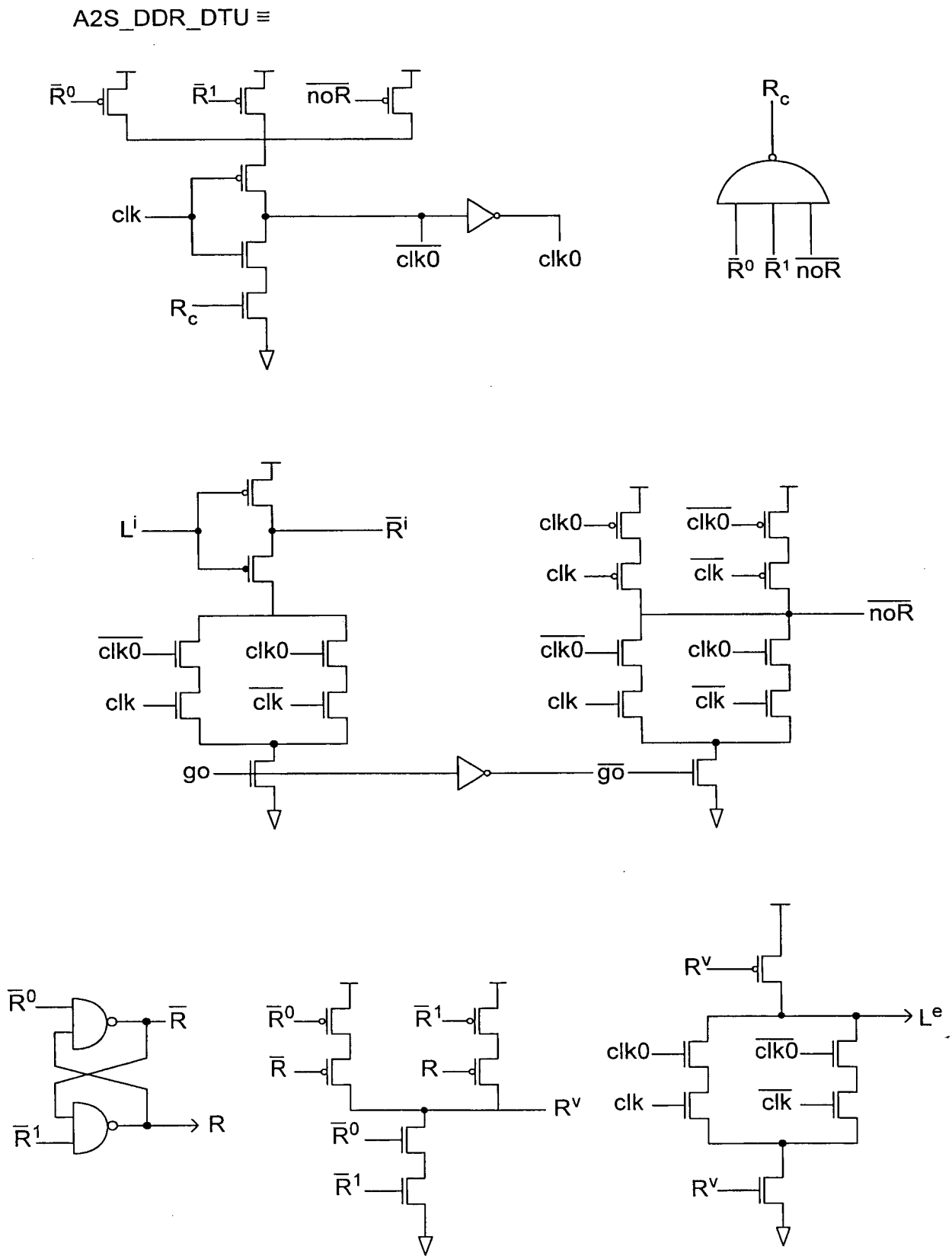

A2S_DDR_DTU ≡
$\bar{R}^0$
$\bar{R}^1$
$\overline{noR}$
clk
$\overline{clk0}$
clk0
R_c
R_c
$\bar{R}^0$ $\bar{R}^1$ $\overline{noR}$
L^i
$\bar{R}^i$
clk0
$\overline{clk0}$
clk
$\overline{clk}$
$\overline{noR}$
go
$\overline{go}$
$\bar{R}^0$
$\bar{R}$
$\bar{R}^1$
R
R^v
L^e

FIG. 63

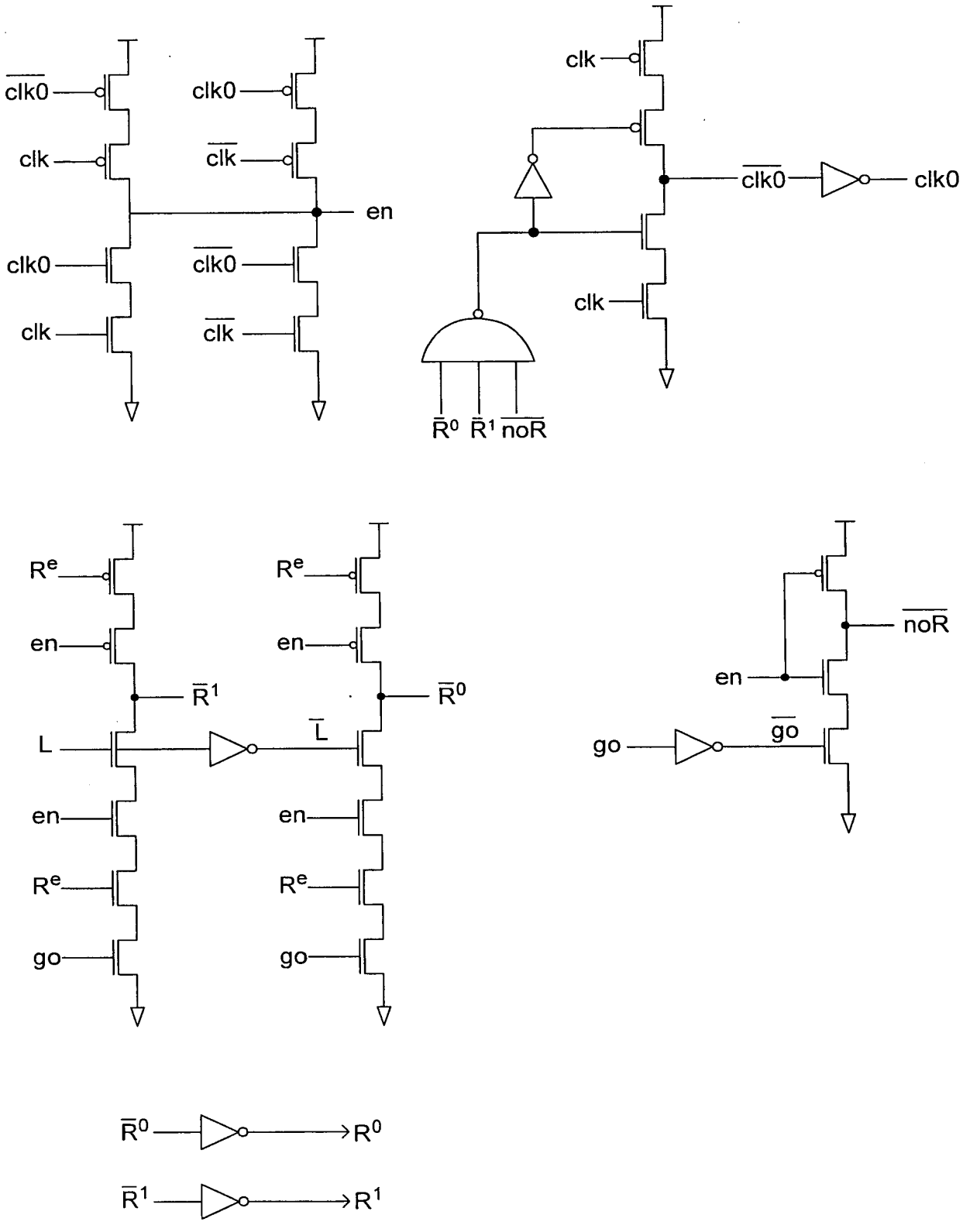
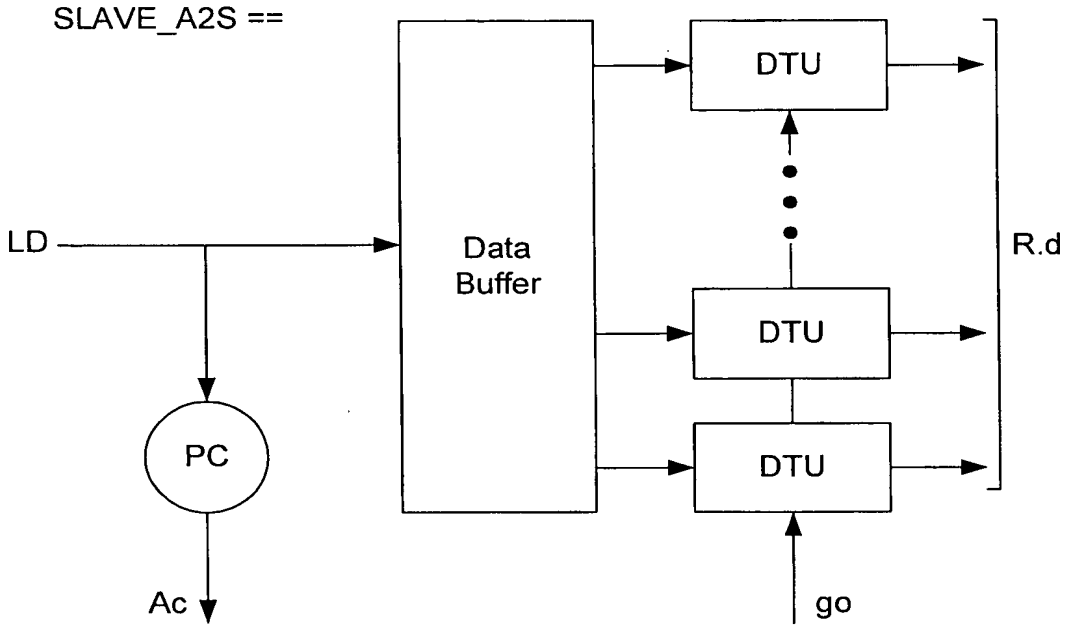

clk0
clk
clk0
clk
en
clk0
clk
clk0
clk
clk
clk0
clk0
clk
$\bar{R}^0$ $\bar{R}^1$ noR
R^e
en
$\bar{R}^1$
L
en
R^e
go
R^e
en
$\bar{R}^0$
$\bar{L}$
en
R^e
go
noR
en
go
go
$\bar{R}^0$
R^0
$\bar{R}^1$
R^1

FIG. 64


SLAVE_A2S ==
LD
Data Buffer
DTU
DTU
DTU
R.d
PC
Ac
go

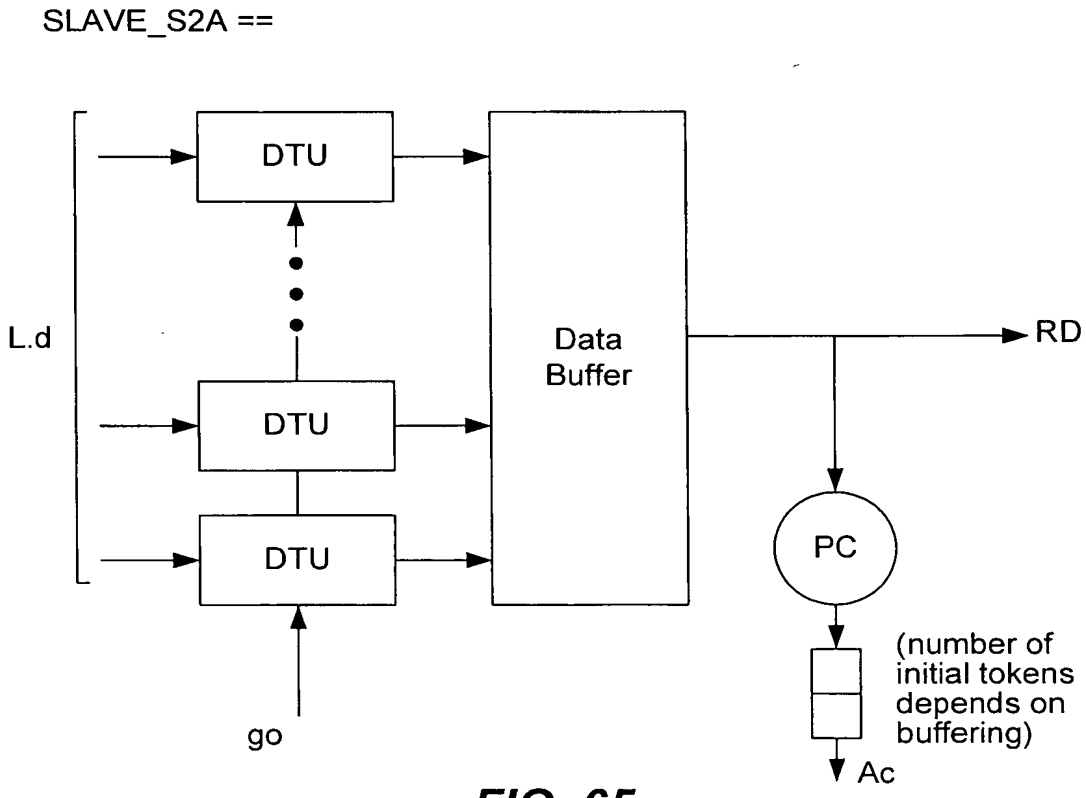

SLAVE_S2A ==
L.d
DTU
DTU
DTU
Data Buffer
RD
PC
(number of initial tokens depends on buffering)
Ac
go

FIG. 65

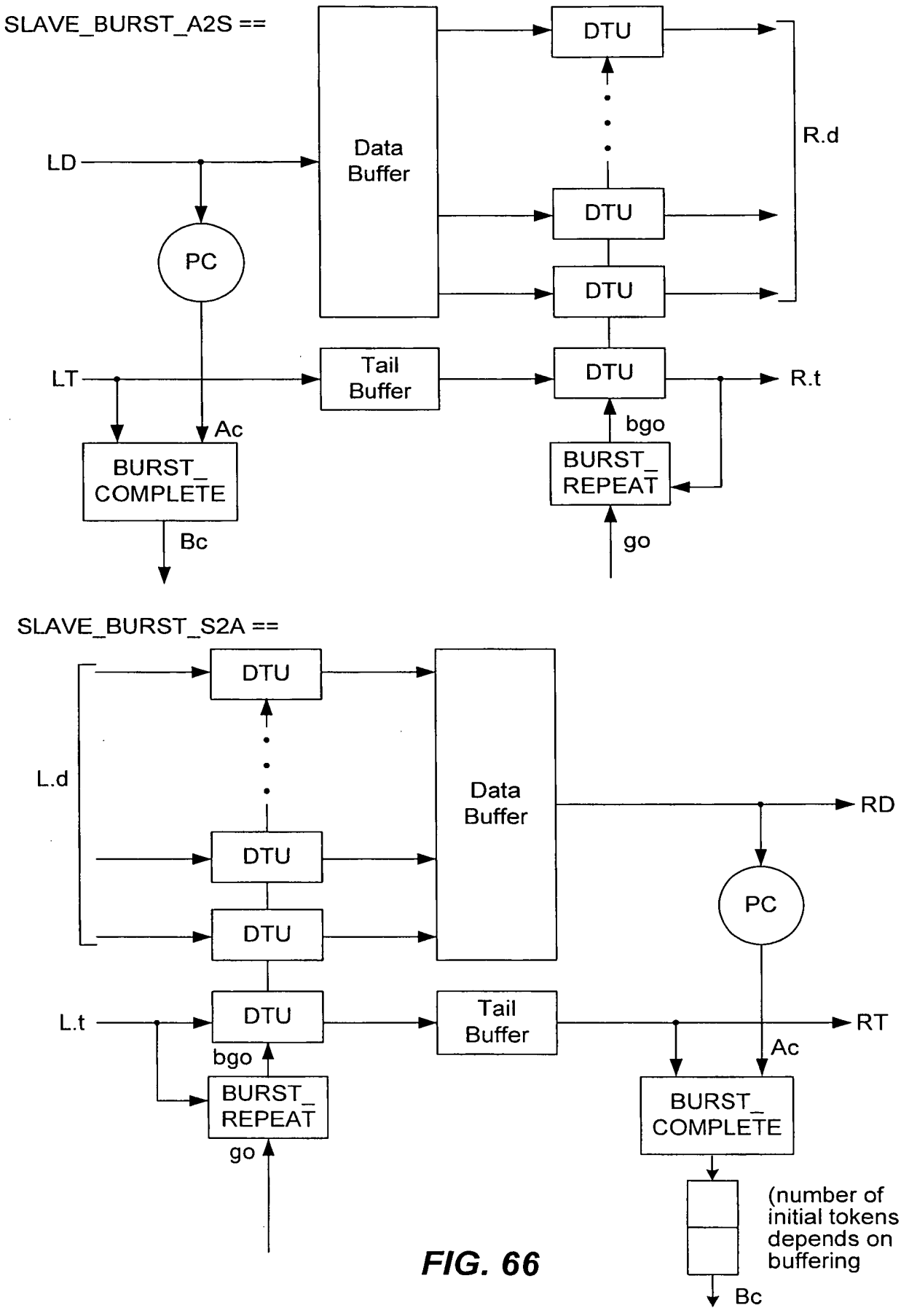

SLAVE_BURST_A2S ==
LD
PC
LT
Ac
BURST_
COMPLETE
Bc
Data
Buffer
Tail
Buffer
DTU
DTU
DTU
DTU
R.d
R.t
bgo
BURST_
REPEAT
go
SLAVE_BURST_S2A ==
L.d
DTU
DTU
DTU
DTU
Data
Buffer
RD
PC
L.t
Tail
Buffer
RT
bgo
BURST_
REPEAT
go
Ac
BURST_
COMPLETE
(number of
initial tokens
depends on
buffering
Bc

FIG. 66

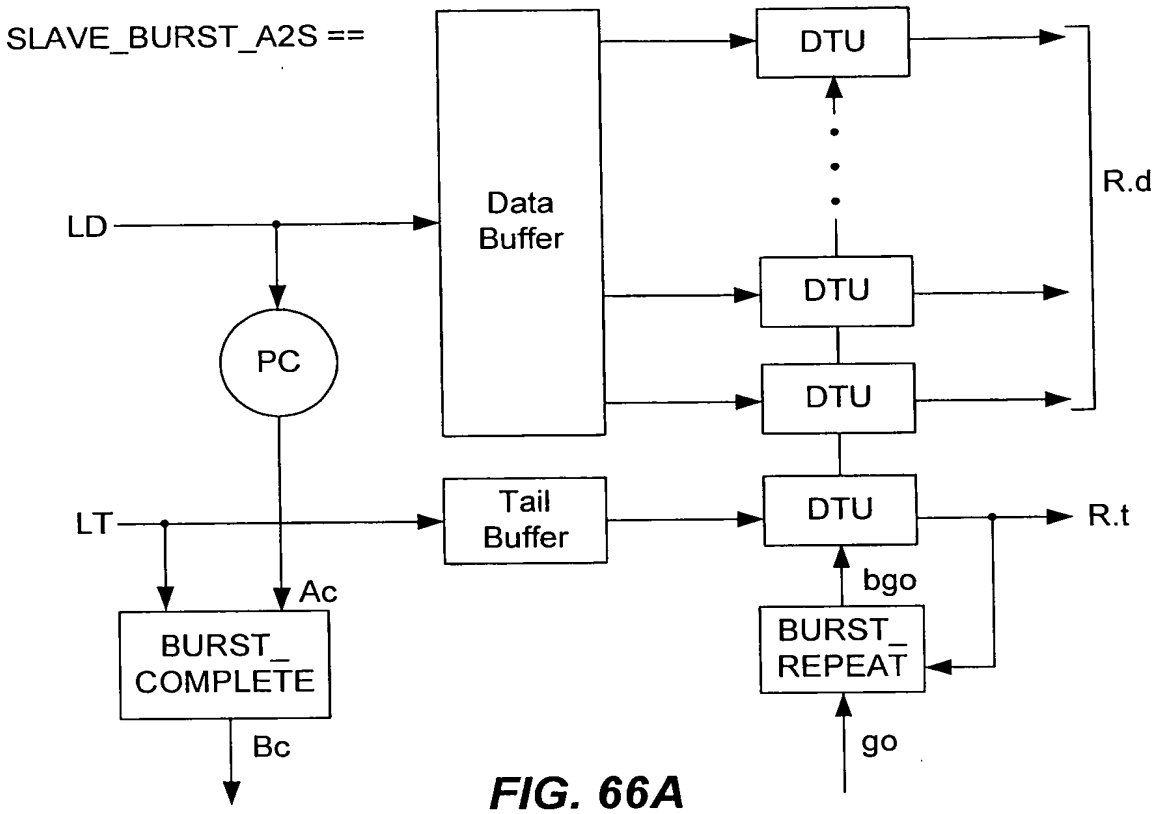


FIG. 66A

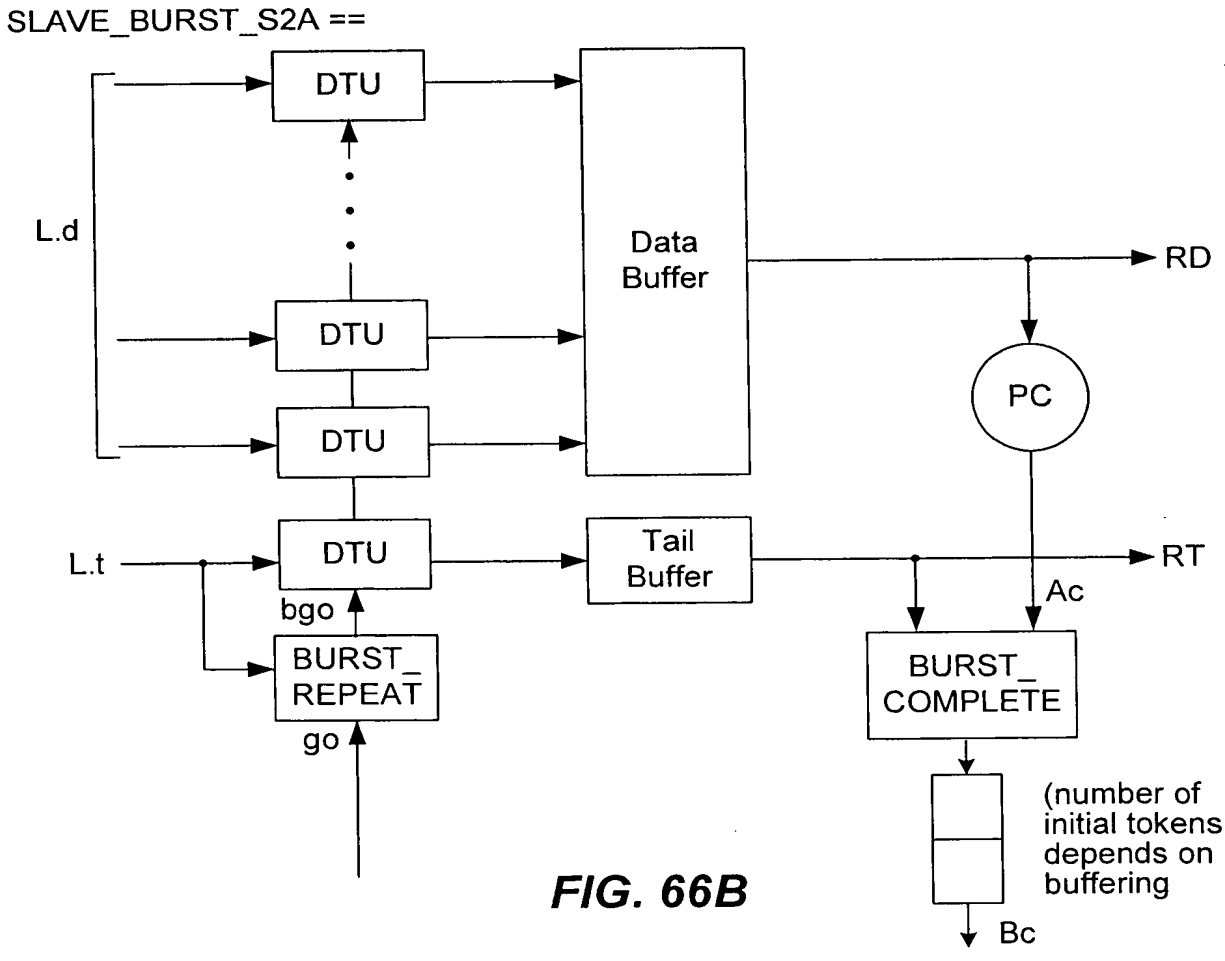


FIG. 66B

SLAVE_FIXED_BURST_A2S ==

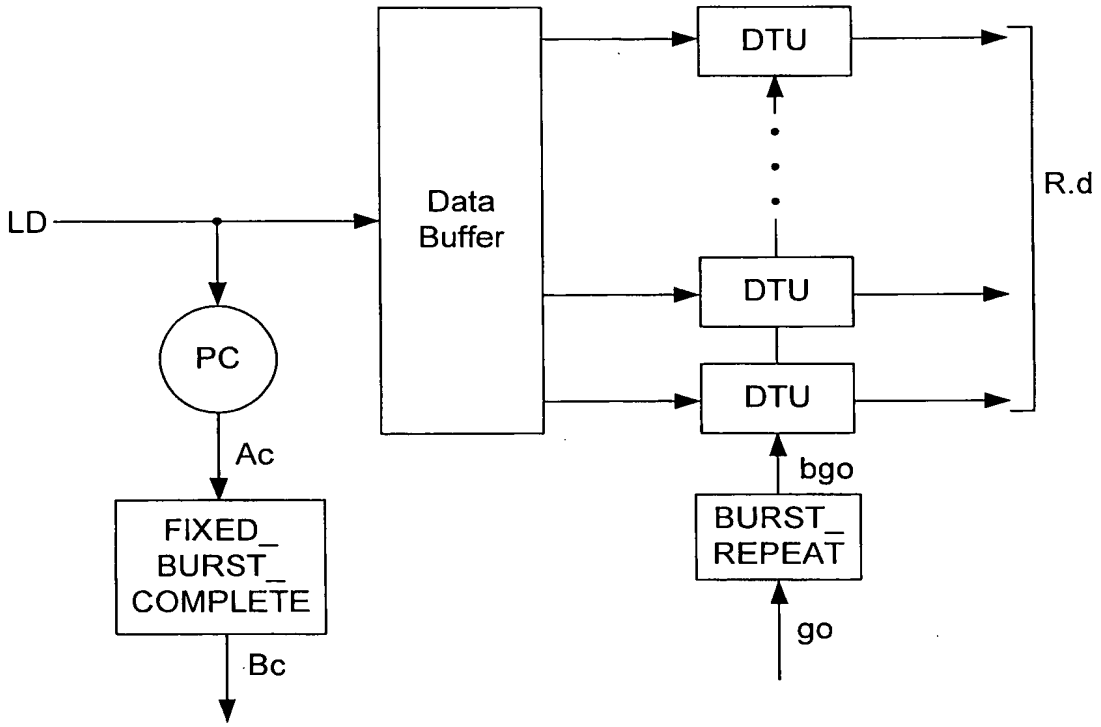


FIG. 67A

SLAVE_FIXED_BURST_S2A ==

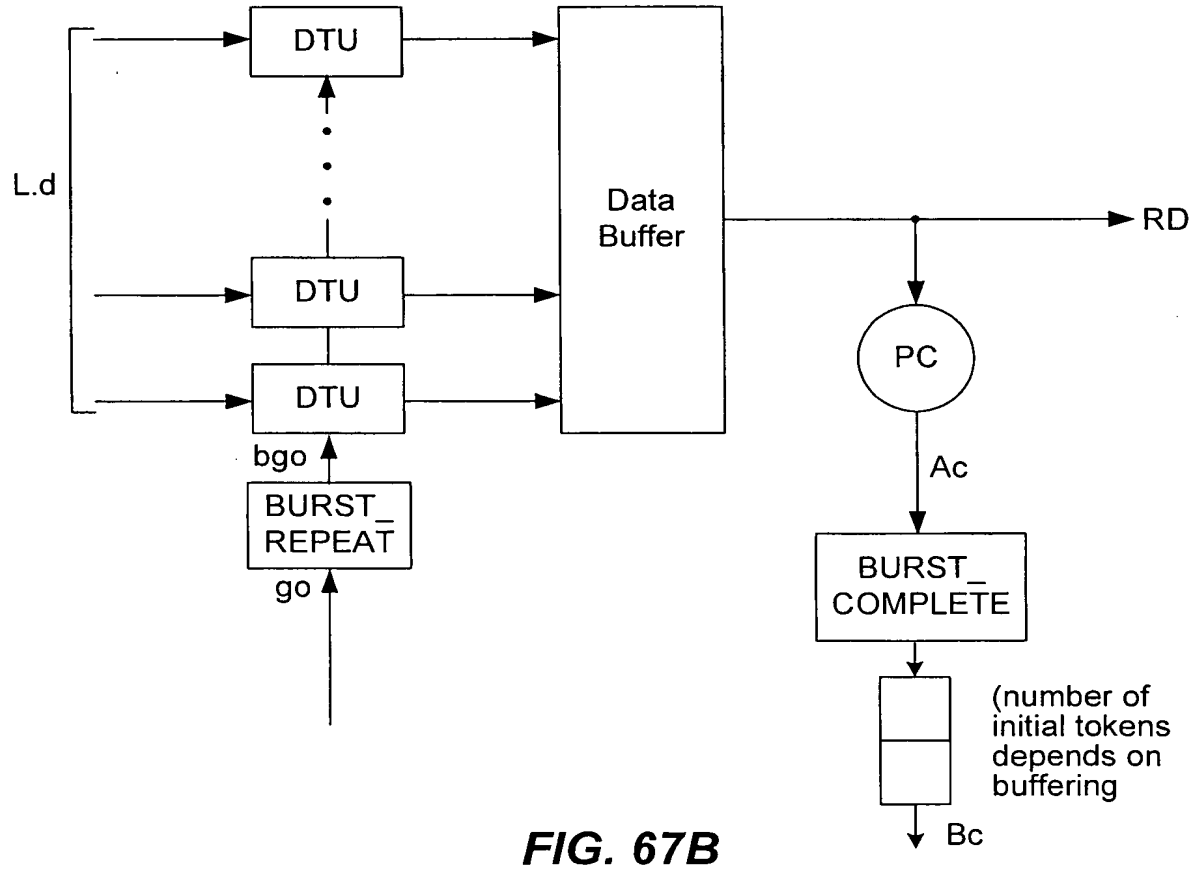


FIG. 67B

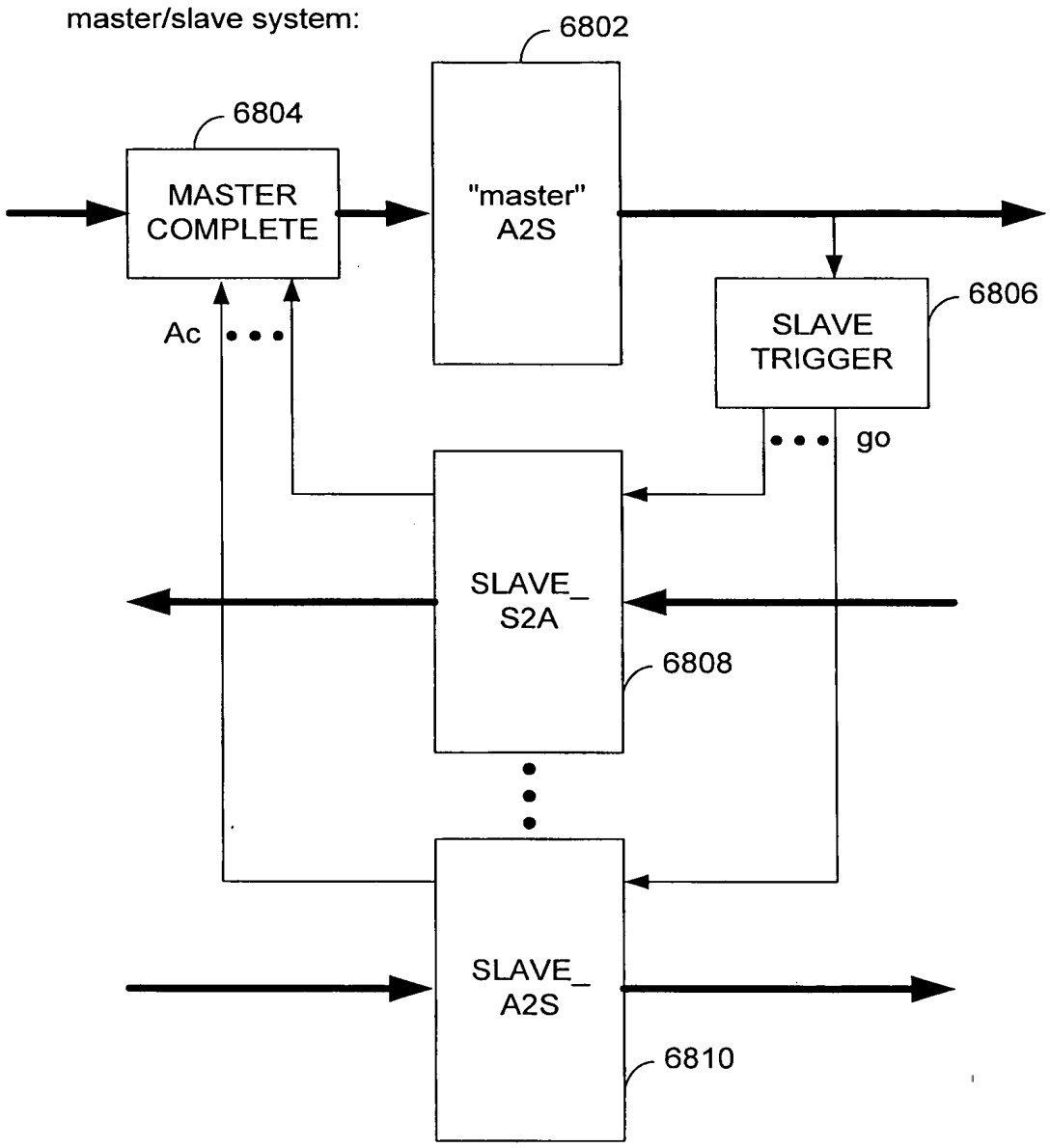

master/slave system:
6802
6804
MASTER COMPLETE
"master" A2S
6806
SLAVE TRIGGER
Ac
go
SLAVE_ S2A
6808
SLAVE_ A2S
6810

FIG. 68

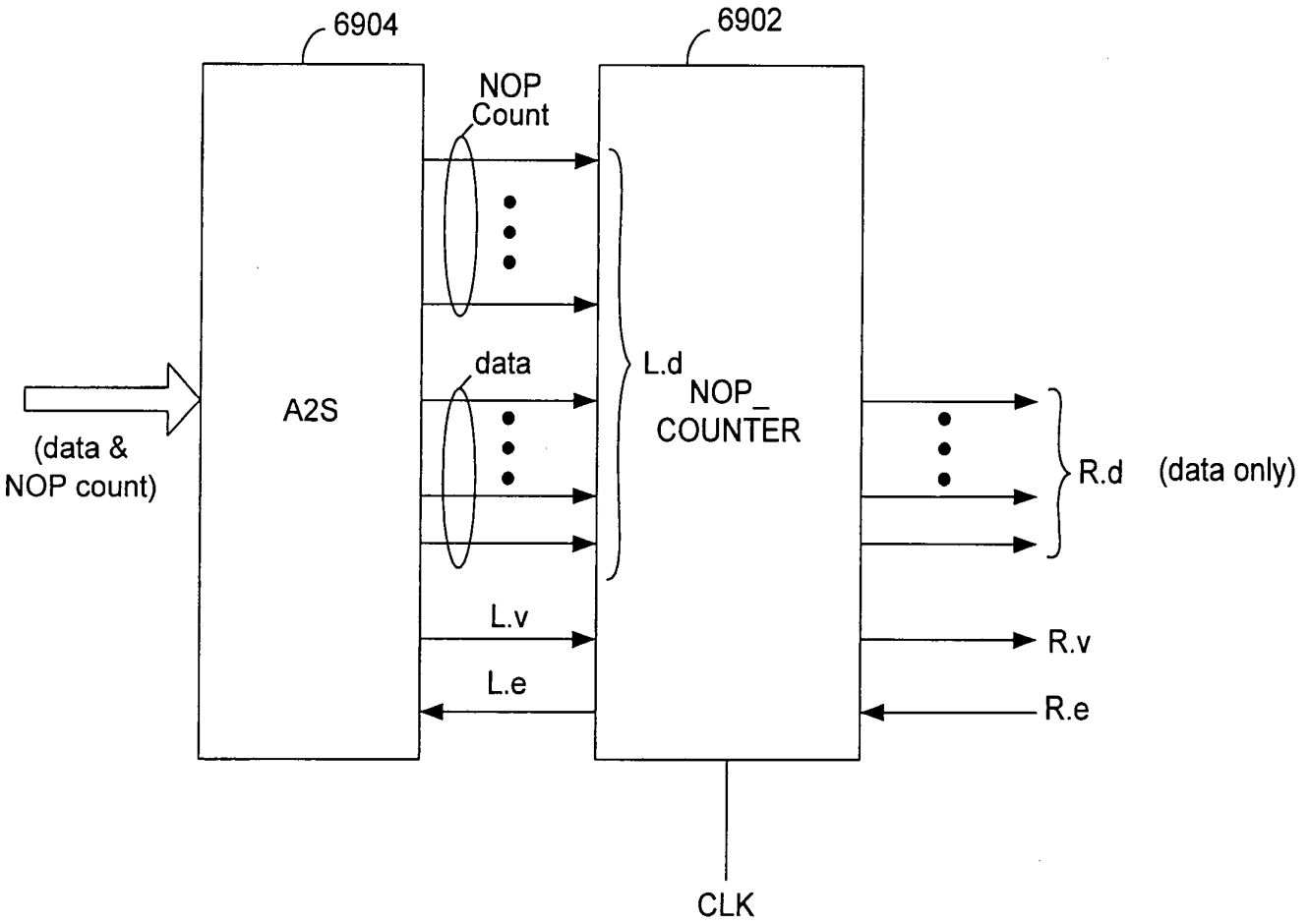

6904
6902
NOP
Count
data
L.d
NOP_
COUNTER
A2S
(data &
NOP count)
R.d
(data only)
L.v
L.e
R.v
R.e
CLK

FIG. 69

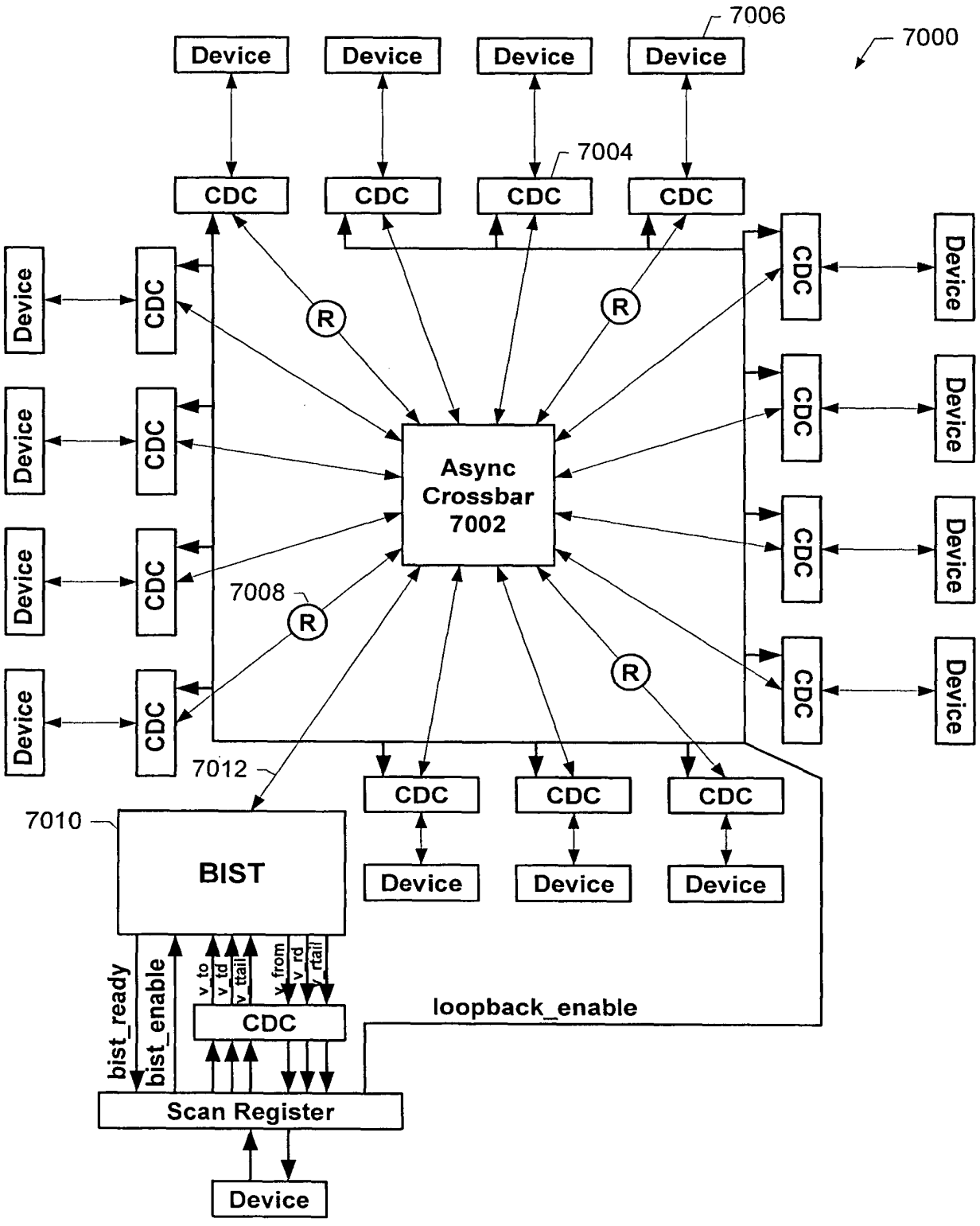

7006
7000
Device
Device
Device
Device
7004
CDC
CDC
CDC
CDC
Device
CDC
R
R
CDC
Device
Device
CDC
Async
Crossbar
7002
CDC
Device
Device
CDC
7008
R
CDC
Device
Device
CDC
R
CDC
Device
7012
CDC
CDC
CDC
7010
BIST
Device
Device
Device
bist_ready
bist_enable
v_to
v_td
v_ttail
v_from
v_rd
v_rtail
CDC
loopback_enable
Scan Register
Device

Fig. 70

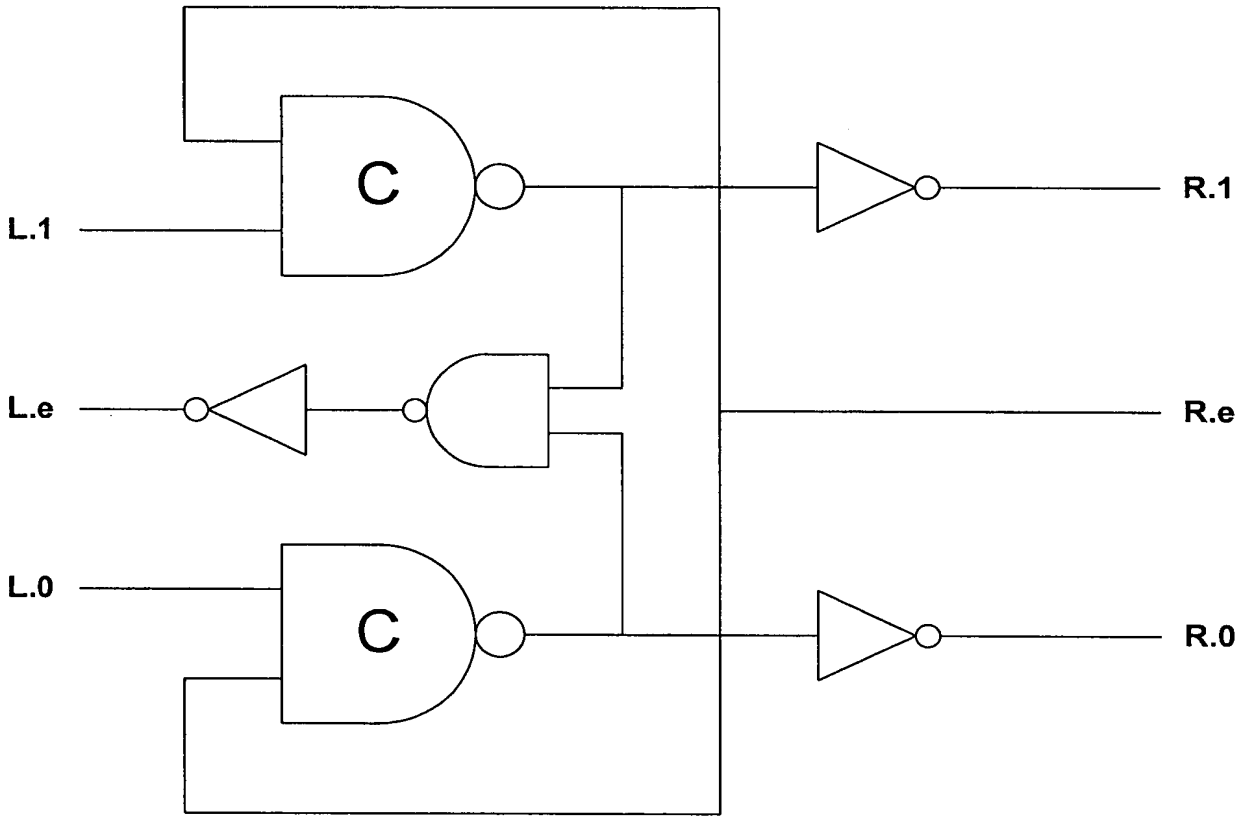


Fig. 70A

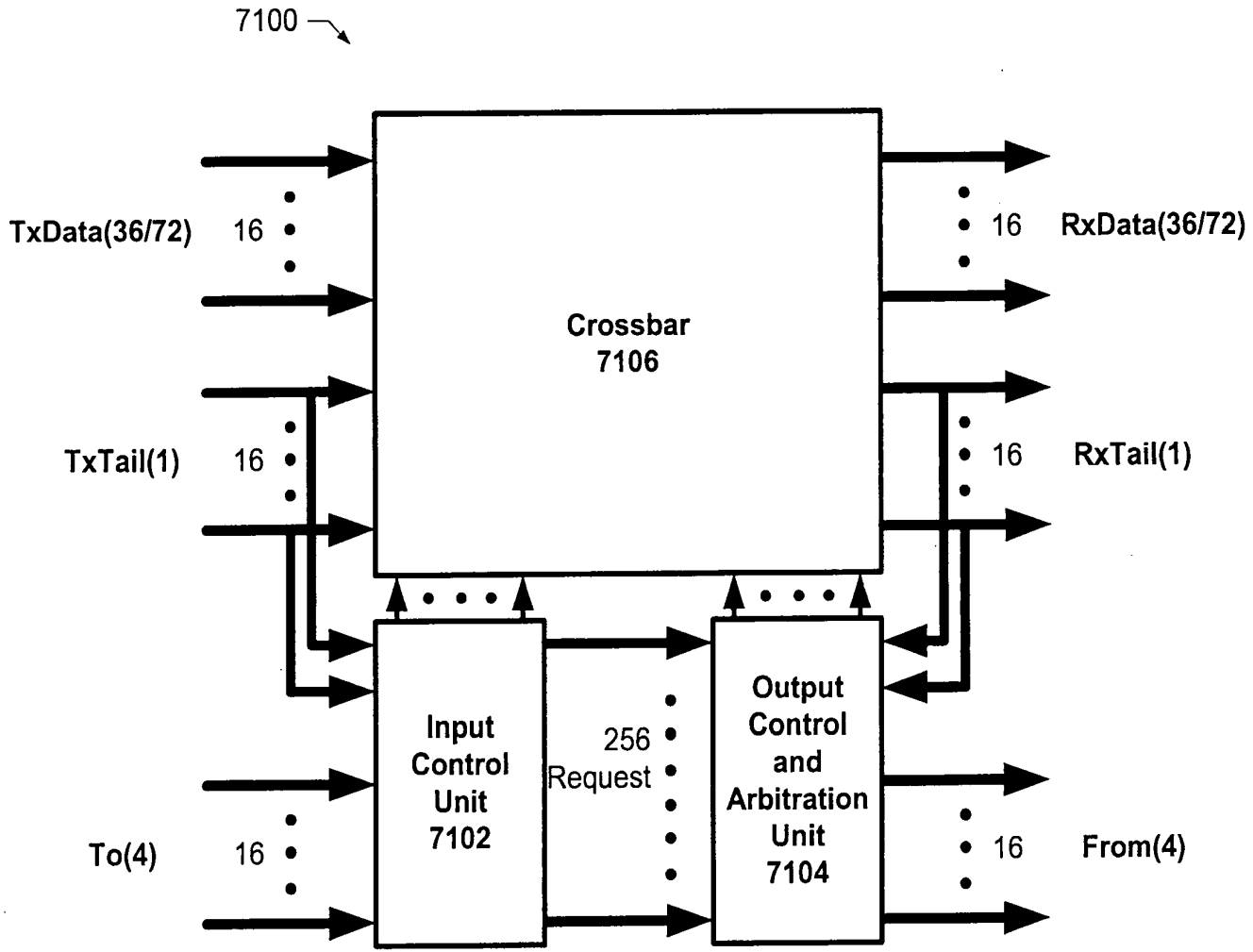

7100
TxData(36/72)
16
TxTail(1)
16
To(4)
16
Crossbar
7106
Input
Control
Unit
7102
256
Request
Output
Control
and
Arbitration
Unit
7104
16
RxData(36/72)
16
RxTail(1)
16
From(4)

Fig. 71

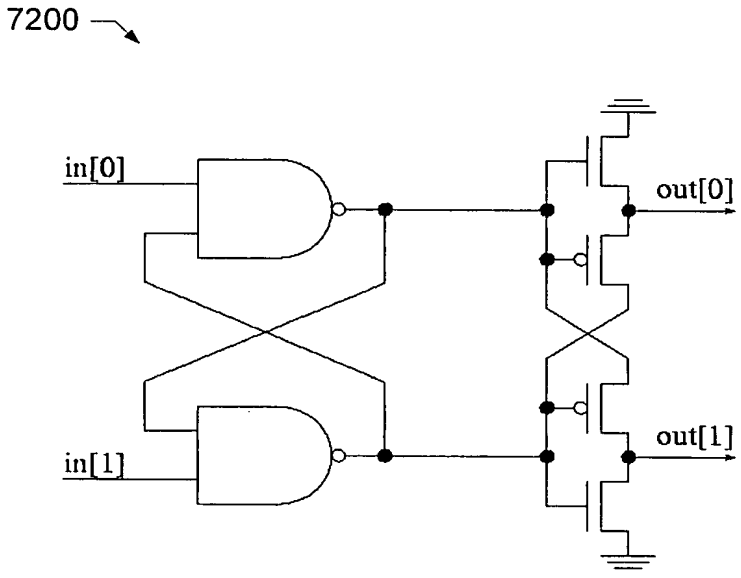


Fig. 72

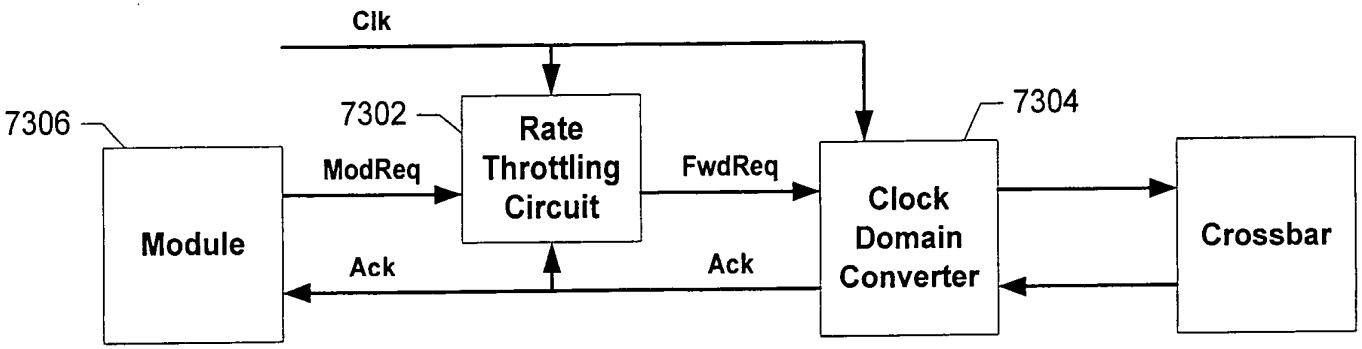

Clk
7306
7302
Rate
Throttling
Circuit
7304
ModReq
FwdReq
Module
Clock
Domain
Converter
Crossbar
Ack
Ack

Fig. 73

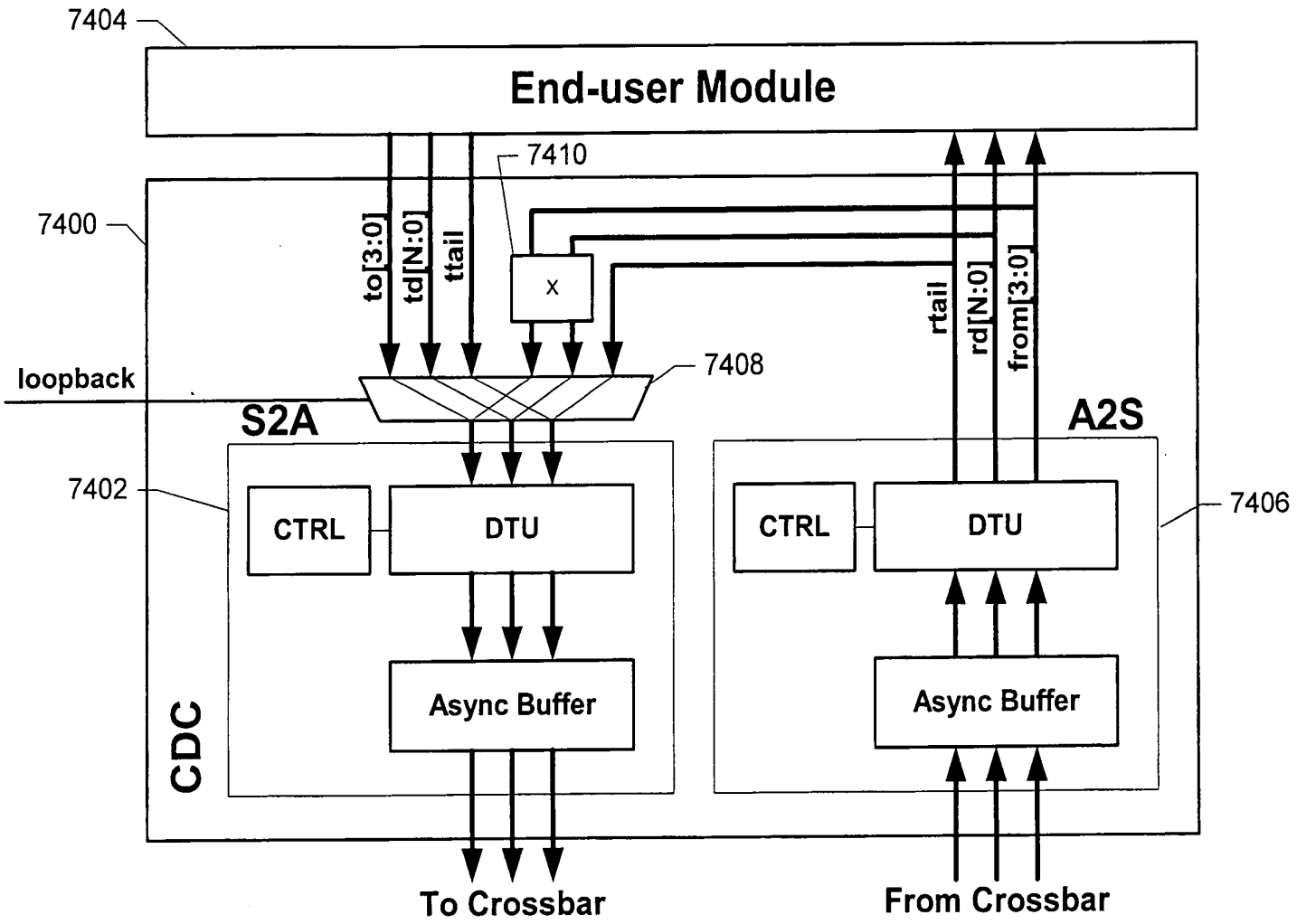

7404
End-user Module
7410
7400
to[3:0]
td[N:0]
ttail
x
rtail
rd[N:0]
from[3:0]
loopback
7408
S2A
A2S
7402
CTRL
DTU
CTRL
DTU
7406
Async Buffer
Async Buffer
CDC
To Crossbar
From Crossbar

Fig. 74

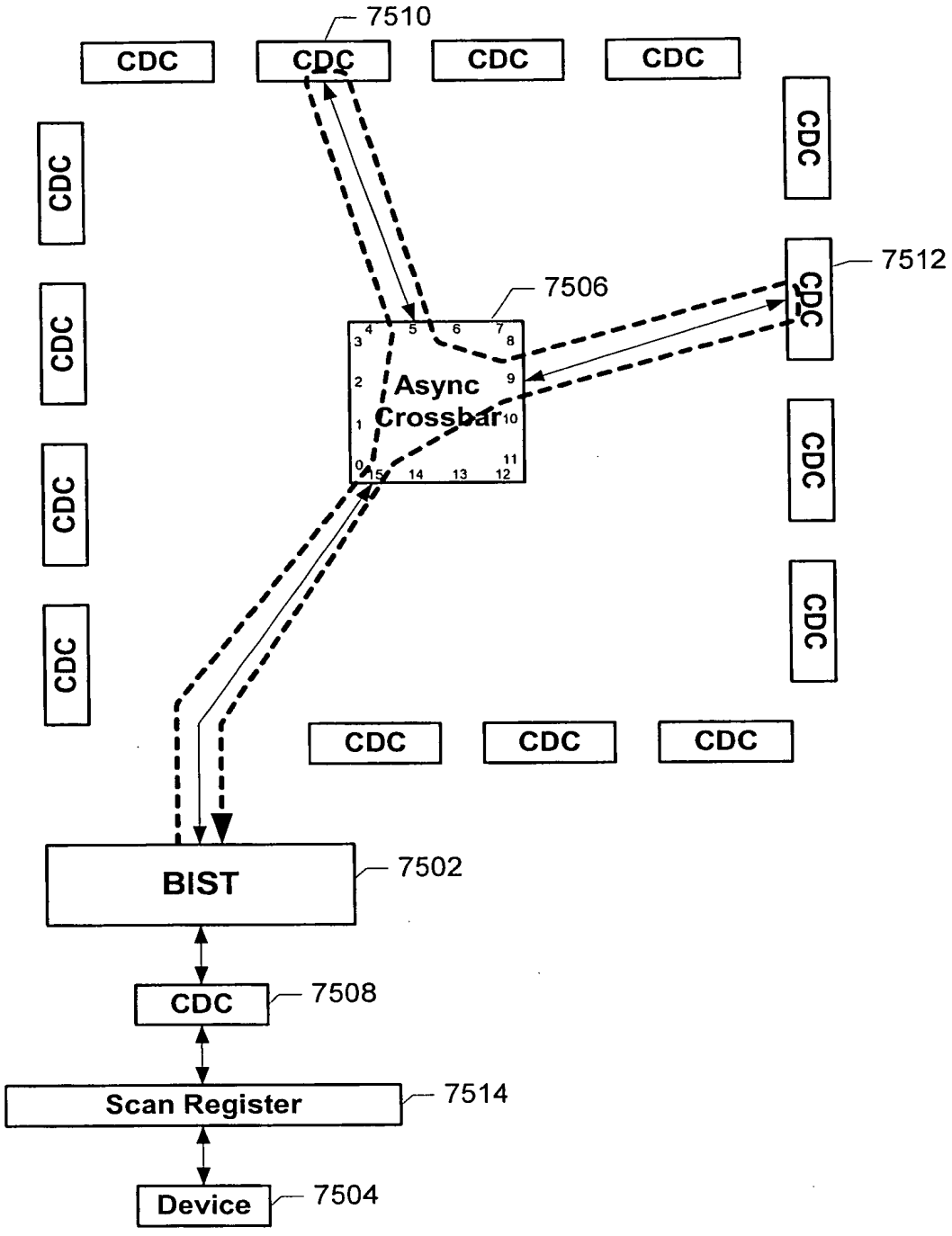

7510
CDC
CDC
CDC
CDC
CDC
CDC
CDC
CDC
CDC
CDC
CDC
CDC
CDC
CDC
7512
7506
Async
Crossbar
BIST
7502
CDC
7508
Scan Register
7514
Device
7504

Fig. 75

Bits	Field	Usage
v_td[3:0]	SecondNode	Defines the second node of the path to test.
v_td[7:4]	Length	Defines the length of the transaction excluding the first word.
v_td[15:8]	Pattern	Defines the 8-bit data pattern used to fill up the transaction.
v_td[31:16]	RepeatCount	The number of times that the same transaction will be sent.
v_td[32:32]	Echo	Forces the BIST to echo the data on v_td, v_to and v_ttail back to the receiver v_rd, v_from and v_rtail respectively without running any test. This is used to test the connection between the scan register and the BIST circuit.
v_td[33:33]	WaitData	Forces the BIST to wait for a transaction before issuing another one.
v_td[35:34]	NotUsed	
v_to[3:0]	FirstNode	Defines the first destination node
v_ttail	Tail	End of transaction.

Fig. 76A

Bits	Field	Usage
v_rd[3:0]	FirstNode	Will contain the first node id.
v_rd[7:4]	Length	Length of the transaction.
v_rd[15:8]	Pattern	8-bit data pattern used to fill up the transaction.
v_rd[31:16]	RepeatCount	Number of times that the same transaction as been sent until an error was encountered.
v_rd[32:32]	Echo	Indicates that the transaction was echoed.
v_rd[33:33]	Pass	Indicates if the test has been executed successfully or not.
v_rd[35:34]	NotUsed	
v_from[3:0]	FirstNode	Defines the first destination node
v_rtail	Tail	End of transaction.

Fig. 76B

72 64	63 56	55 48	47 40	39 32	31 24	23 16	15 8	7 0
X	X	X	X	X	X	X	X	nextnode
pattern	pattern	pattern	pattern	pattern	pattern	pattern	pattern	pattern
pattern	pattern	pattern	pattern	pattern	pattern	pattern	pattern	pattern
...	...	...	...	...	...	...	...	...
pattern	pattern	pattern	pattern	pattern	pattern	pattern	pattern	pattern

Fig. 76C

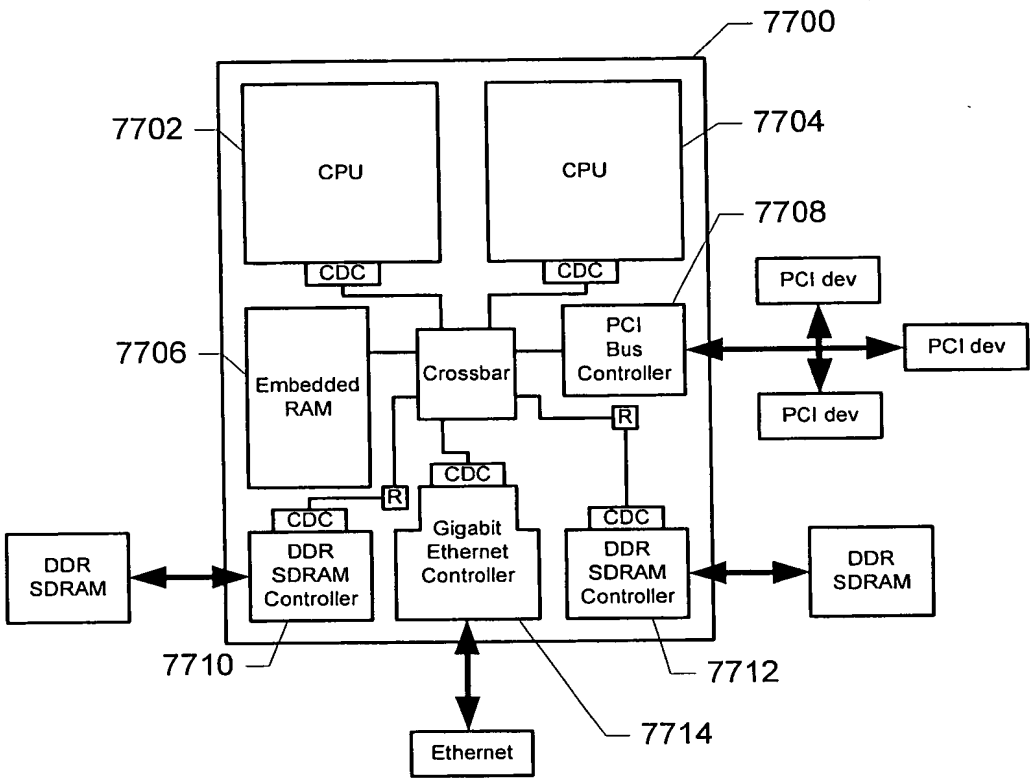

7700
7702
CPU
CPU
7704
7708
CDC
CDC
PCI dev
7706
Embedded RAM
Crossbar
PCI Bus Controller
PCI dev
PCI dev
R
CDC
R
CDC
CDC
DDR SDRAM
DDR SDRAM Controller
Gigabit Ethernet Controller
DDR SDRAM Controller
DDR SDRAM
7710
7712
7714
Ethernet

Fig. 77

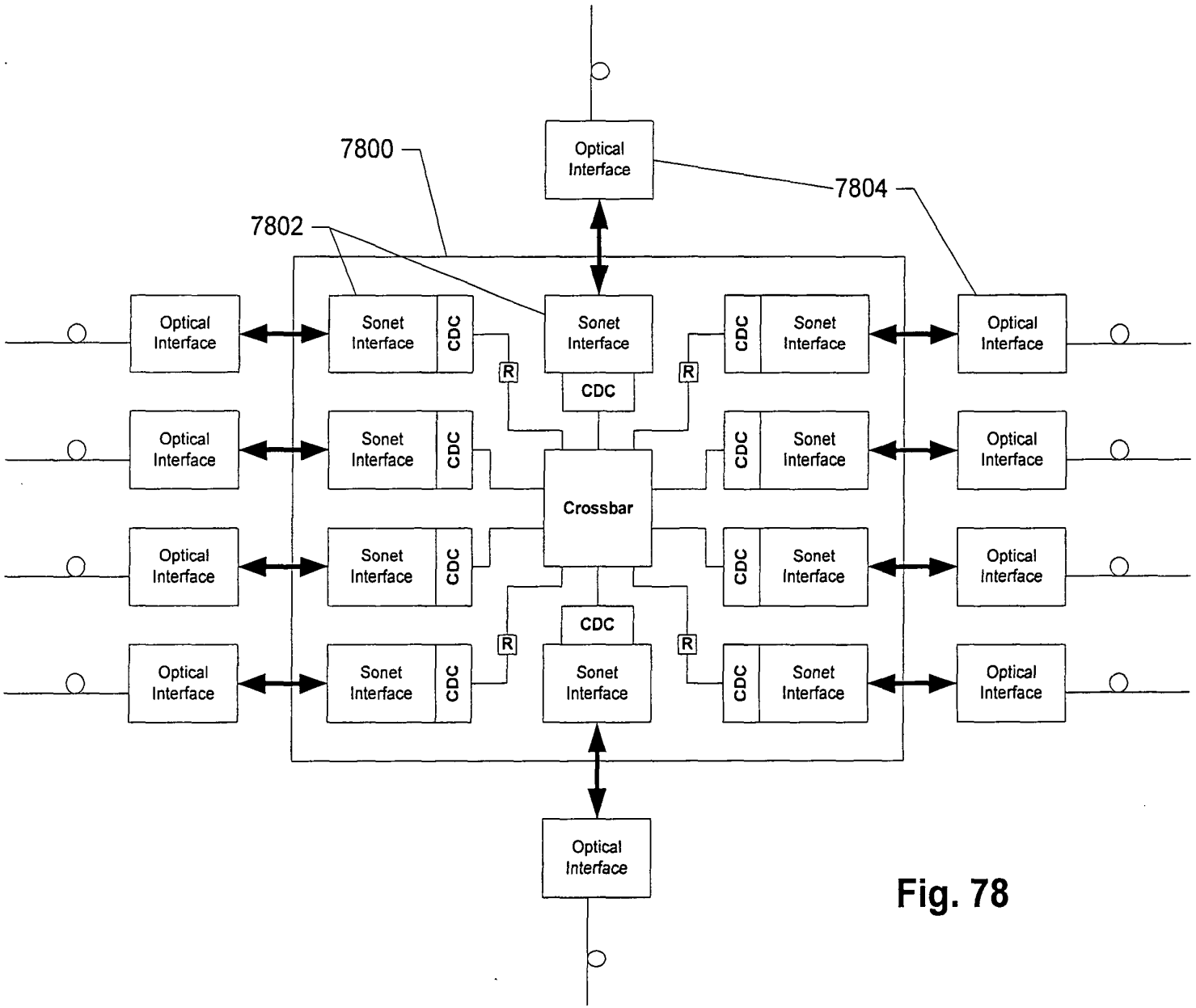

7800
7802
7804
Optical Interface
Sonet Interface
CDC
R
Crossbar

Fig. 78